FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING

DEGREE PROGRAMME IN WIRELESS COMMUNICATIONS ENGINEERING

MASTER'S THESIS

EVALUATION OF MACHINE LEARNING TECHNIQUES FOR INTRUSION DETECTION IN SOFTWARE DEFINED NETWORKING

Author	Ahnaf Ahmad
Supervisor	Mika Ylianttila
(Technical Advisor	Ijaz Ahmad)

June 2020

Ahmad A. (2020) Evaluation of Machine Learning Techniques for Intrusion Detection in Software Defined Networking. University of Oulu, Faculty of Information Technology and Electrical Engineering, Degree Programme in Wireless Communications Engineering. Master's Thesis, 50 p.

ABSTRACT

The widespread growth of the Internet paved the way for the need of a new network architecture which was filled by Software Defined Networking (SDN). SDN separated the control and data planes to overcome the challenges that came along with the rapid growth and complexity of the network architecture. However, centralizing the new architecture also introduced new security challenges and created the demand for stronger security measures. The focus is on the Intrusion Detection System (IDS) for a Distributed Denial of Service (DDoS) attack which is a serious threat to the network system. There are several ways of detecting an attack and with the rapid growth of machine learning (ML) and artificial intelligence, the study evaluates several ML algorithms for detecting DDoS attacks on the system.

Several factors have an effect on the performance of ML based IDS in SDN. Feature selection, training dataset, and implementation of the classifying models are some of the important factors. The balance between usage of resources and the performance of the implemented model is important. The model implemented in the thesis uses a dataset created from the traffic flow within the system and models being used are Support Vector Machine (SVM), Naive-Bayes, Decision Tree and Logistic Regression. The accuracy of the models has been over 95% apart from Logistic Regression which has 90% accuracy. The ML based algorithm has been more accurate than the non-ML based algorithm. It learns from different features of the traffic flow to differentiate between normal traffic and attack traffic. Most of the previously implemented ML based IDS are based on public datasets. Using a dataset created from the flow of the experimental environment allows training of the model from a real-time dataset. However, the experiment only detects the traffic and does not take any action. However, these promising results can be used for further development of the model.

Key words: Software Defined Networking (SDN), Intrusion Detection System (IDS), Distributed Denial of Service (DDoS), Machine Learning (ML), OpenFlow, Mininet, POX Controller.

TABLE OF CONTENTS

FOREWORD

At first, I want to thank the almighty Allah, for helping me throughout the period and completing it successfully. This thesis marks an interesting journey in my life. It bears the memories of my life in Finland. I have learned and grown so much during this period. The knowledge, skills and memories throughout this time will be a big part of my life.

This master's thesis has been done at the Centre for Wireless Communications at the University of Oulu according to the requirements of the master's degree programme of Wireless Communications Engineering under the supervision of Prof. Mika Ylianttila. I want to thank him for his support, his supervision, and the chance to work with him. I also want to express my gratitude towards my technical supervisor Ijaz Ahmad. His constant support and supervision throughout the time helped me a lot in finishing this work. I want to thank him for pushing me to grow and learn more. The thesis work aims to evaluate different machine learning techniques for intrusion detection system in an SDN environment.

This marks the end of my master's journey at the University of Oulu. I am thankful and grateful to the people whom I met during this period and who have become a big part of my life. I want to thank my friends for supporting me, encouraging me, and helping me to see my potential. The overall university life has been amazing with great study opportunities and facilities that the university offers. The University of Oulu will have a special place in my heart.

I want to mention and thank the most important people in my life, my family. They have always been my pillar to hold on to in any situation of my life. My mother and father have always supported me in every possible way for which I cannot thank them enough, and I am forever grateful towards them. I am thankful to my sisters for always being there, motivating me and encouraging me. Thank you, to all my family members for supporting, believing in, and loving me.

Oulu, 16 June 2020

Ahnaf Ahmad

LIST OF ABBREVIATIONS AND SYMBOLS

AA	Active Application
ABA	Anomaly Based Approach
ANN	Artificial Neural Network
API	Application Programming Interface
AS	Autonomous System
ASIC	Application Specific Integrated Circuit
AUC	Area Under Curve
BBF	The Broadband Forum
BGP	Border Gateway Protocol
CPU	Central Processing Unit
DDoS	Distributed Denial of Service
DoS	Denial of Service
EE	Execution Environment
FAWG	Forwarding Abstractions Working Group
FN	False Negative
FP	False Positive
ForCES	Forwarding and Control Element Separation
GUI	Graphic User Interface
I2RS	Interface to the Routing System
IETF	Internet Engineering Task Force
IDS	Intrusion Detection System
IP	Internet Protocol
IoT	Internet of Things
IPS	Intrusion Prevention System
KNN	k-Nearest Neighbours
MEF	Metro Ethernet Forum
ML	Machine Learning
NBI	Northbound Interface
NDM	Negotiable Data Path Model
NOX	Network Operating System
OIF	Optical Internet Forum
ONF	Open Network Foundation
ONOS	Open Network Operating System
ONS	The Open Networking Summit
POF	Protocol-oblivious forwarding
POX	Pythonic Networking Operating System
QoS	Quality of Service
RAM	Random Access Memory
RBF	Radial Based Function
REST	Representational State Transfer
ROC	Receiver Operating Characteristics

RPC	Routing Control Platform
SBI	Southbound Interface
SBA	Signature Based Approach
SDN	Software Defined Networking
SDNRG	Software-Defined Networking Research Group
SOM	Self Organizing Map
SSD	Solid State Drive
SSID	Service Set Identifiers
SSL	Secure Socket Layer
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
TSL	Transport Security Layer
UDP	User Datagram Protocol

1 INTRODUCTION

Connectivity and distribution services across the globe have grown rapidly through the Internet. This rapid growth has ushered in a new era of connectivity and control of the Internet. The increase of Internet usage means the increase of data including personal, commercial, military and government information. The growth of this data also increases security threats. The number of data breaches doubled from 2015 to 2017 alone [1], and it is ever increasing. Internet connectivity and control has been manual for a very long time until Software Defined Networking (SDN) was introduced into the network system [2]. This scheme was introduced to simplify network development for this ever growing and important aspect of global connectivity. This gives an opportunity to manage the network in a much simpler way compared to the previous methods.

The concept of SDN pushes the network system to more software-based control than hardware-based. It mainly separates the control and data planes and it is logically centralized. This concept split the network intelligence from packet switching to a centralized controller and the packet forwarding decisions are managed from the controller to the switches through standard protocol such as OpenFlow [2]. OpenFlow is a standard protocol which allows the control plane to communicate with the switches. OpenFlow is not the only protocol for SDN but one of the most popular. The SDN architecture is generally composed of three main layers: Infrastructure Layer, Control Layer and Application Layer. The infrastructure layer is composed of different elements that form an underlying network to forward network traffic. The most important layer in the SDN architecture is the control layer, which is responsible for traffic engineering, traffic management, network management and so on. The application layer allows the creation of policies, network management rules, and Quality of Service (QoS) for the controller. Two protocols are associated with the communication between the layers. The Northbound Interface (NBI) is the protocol for communication between the controller and the application layer. The Southbound Interface (SBI) provides communication between the controller and the infrastructure layer.

SDN paved a new way to approach the network in a much easier and more convenient way. Nevertheless, new ways of attacking a network system came along and with the SDN architecture, the effects can be catastrophic. Different types of attacks can be implemented on the SDN architecture, such as network manipulation, traffic diversion, side channel attack, Denial of Service (DoS) and so on. The focus of this thesis is Intrusion Detection System (IDS), and it works by overviewing the traffic, analysing it and trying to find anomalies or unauthorized access in the network domain. Sometimes IDS can take necessary measures to secure the network. There are several issues regarding the IDS as it constantly measures the network traffic and uses a lot of resources even if there is no attack. Even though the traffic is monitored constantly, the amount of time to respond is quite significant and the network administrator needs to constantly provide updates to the protection mechanism. Otherwise, the system becomes vulnerable as time passes by.

Since. SDN's centralized network architecture makes it vulnerable to numerous attacks, it also paves the way to make a more efficient detection system and to implement it [3]. Distributed Denial of Service (DDoS) has been one of the major attacks in the past and it is constantly growing with new ways to attack the system. As the Internet is surging ahead, the attack system is also growing and making more and more hosts vulnerable to these attacks. Several features have been used for detecting attacks in SDN. High traffic rate and deviation from the normal traffic rate in the flow have been two of the important features to detect DDoS attack. Most of the IDS in SDN are based on these two features [4].

Machine Learning (ML) based IDS has been on the rise for the past few years and developers are trying to find out suitable and better ML methods and ways to implement it in the network. The main concern is the efficiency, training of complex models, and the amount of different data. It uses statistical methods to train the data and make prediction based on the training. This thesis evaluates different ML algorithms in the SDN environment using a unique dataset with a module that works with the centralized controller. The focus of the experiment is to implement a simple ML-based detection system in an SDN controller. The ML algorithms used are, Support Vector Machine (SVM) [5], Naive-Bayes [6], Decision Tree [7] and Logistic Regression [8]. The aim is to have a detection system which learns from real-time data. The comparison of the models from different test results provides a brief idea about the algorithms regarding the dataset, feature sets, a way of implementation, and how to improve.

The thesis organization is as follows. Chapter 2 discusses the main element of the thesis work, which is the SDN, the discussion includes the history, the architecture, and the elements of SDN. Chapter 3 focuses on network security, different security threats, defence techniques and specific security threats in SDN. Chapter 4 discusses work related to the approach of IDS in SDN. Chapter 5 provides the proposed method of the work and the elements which are used for the thesis. Chapter 6 has the simulation and results of the thesis work. Chapter 7 includes the discussion of the work and Chapter 8 concludes the whole work.

2 SOFTWARE DEFINED NETWORKING

The growth of the Internet as well as the growth of the networking system has been a transforming event. It improved people's lives in so many ways. Not only from the technological aspect but also socially and economically [9]. Network system architecture has not been at the same pace of advancement as the services it is providing. The conventional hardware-based network architecture is based on a special algorithm dedicated to the devices. The architecture, routing path, data flow of the network, and the algorithms are specific to the hardware system [2]. The computer networks being complex and these networking algorithms also being hardware-specific, makes it difficult to manage the entire system because of its algorithm being network or hardware-specific [10]. The conventional system is more vendor-specific and the network administrator needs to configure the network devices separately with individual configuration interfaces. Even though some network vendors do give some central management, it is restricted to their own protocols, configuration, and mechanism [10]. These drawbacks have hampered the innovation and progress of network systems. However, they did not decrease the operation complexity of network management work. It delays the reaction towards different network events such as intrusion detection, traffic shifting, and so on. With the rapid progress of the sophisticated policies of networks and the complexity of the tasks that need to be done, the need for a new and better way of handling these events has developed [11].

Nowadays, network operators are in need of a faster and easier way to manage their networks. Legacy IP networks have complexity, which is difficult to manage, and, moreover, it is vertically integrated which means that the control and data planes are merged together. This architecture presented a need for a new way of integration and managing the network [10]. SDN provides the physical separation of the data plane and control plane, with the control plane having the overall control of the network system. SDN is dynamic, manageable, adaptive, and paves a way to solve the network architecture problems. It provides the platform to have centralized control of the overall network behaviour [10]. SDN is shifting the way to a much easier and simpler way of managing networks by centralizing control through the control plane. The control plane dictates the network's data plane components such as routers, switches, and different middleboxes with a well-defined Application Programming Interface (API) [12]. As SDN decouples the control plane, it paves the way for innovation through the programmability of the forwarding devices as well as the control plane [12]. The core difference is that the control plane is a software component running on top of the hardware and it is easily programmable. This provides the opportunity for new ideas and the operators do not need to configure every component to make some changes in the network behaviour.

2.1 History

The introduction of the SDN concept came along in 2010, as a new paradigm of the networking system aiming for easing the control and management of the network environment [10]. However, the evolution towards SDN started with programmable networks and the timeline can be pushed back as far as the mid-1990s. It started with the introduction of the programmable function in the network system to reduce the barrier for deploying new services [10]. The Internet was taking off fast during the mid-1990s. It paved the way for new services and drew the attention of researchers to innovate and work on new ideas. An approach of a re-programmable network system came to light and was termed active networking. It opened up a vision for a programmable interface. The research program worked on alternatives from the

traditional network system. The focus of the active network research was to specify the functional components of a network node instead of end-to-end services. The target was to set up the right programmable node which would pave the way for supported end-to-end services [13]. The components from a single node are shown in Figure 1 where each node is capable of running one or more execution environments (EEs) and users are able to invoke Active Applications (AAs) which give the code for the EEs to allow an end-to-end service.

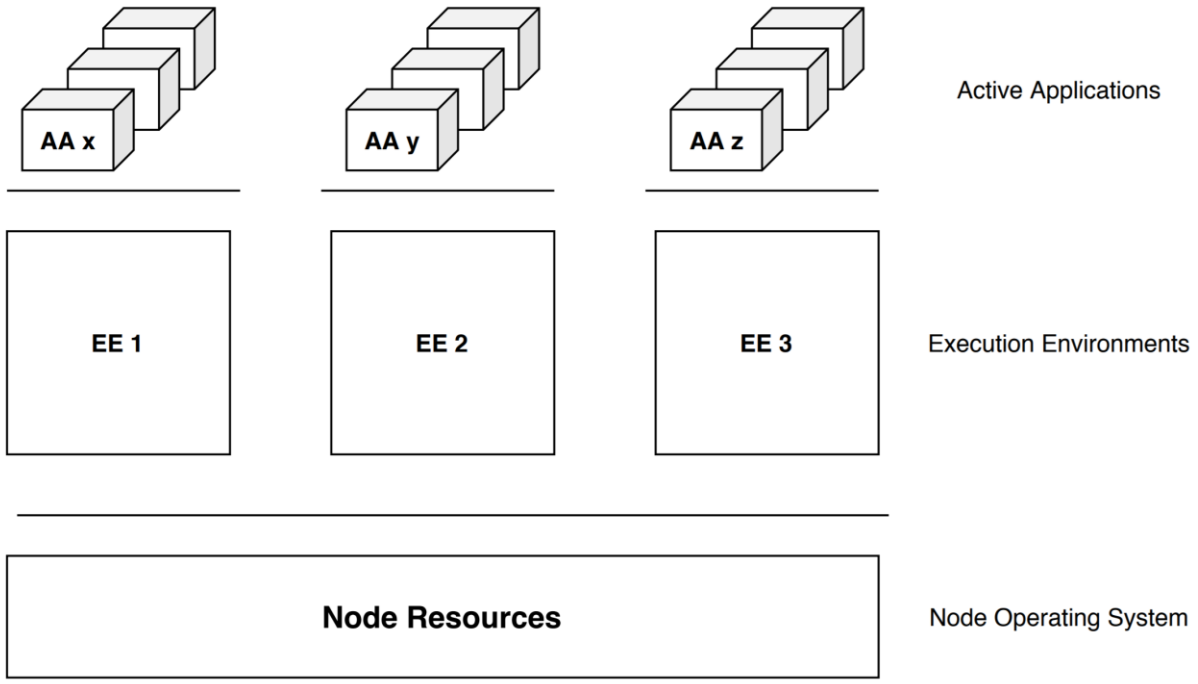

Figure 1. Active Node Architecture.

In the early 2000s, an increase in traffic and more focus on network reliability and performance created a need for controlling certain network features such as the path of the packets which is also known as traffic engineering. Implementation was difficult as the routers and switches are tightly coupled with each other. For this reason, certain tasks regarding controlling the traffic, routing behaviour, and so on was challenging. This led to the emergence of the separation of control and data planes [10]. This followed the trend of two main innovations, an open interface between control and data planes such as ForCES (Forwarding and Control Element Separation) [14] and logically centralized control of the network such as Routing Control Platform (RPC) [15].

While exploring possible formats for a central network control system for the networks, back in 2004, the concept of RCP, as shown in Figure 2 emerged as a solution for the interdomain routing control for IP-networks. It calculates the Border Gateway Protocol (BGP) routes for the Autonomous System (AS) from a central server that provides higher flexibility and control for the network operators for BGP routing control [16]. The RPC is responsible for selecting the routing path on behalf of the IP routers. The circles in Figure 2 represent conventional routers. Also, RPC has the option to exchange information with other domains. There have been several benefits of the RCP including control over protocol interactions, network-wide path selection and policy, and redefinition of inter-AS routing. The platform focuses on robustness, scalability, convergence speed, and transient consistency to avoid the challenges it can face as

a single point working node. This method addresses the advantages and challenges of a single point control [16].

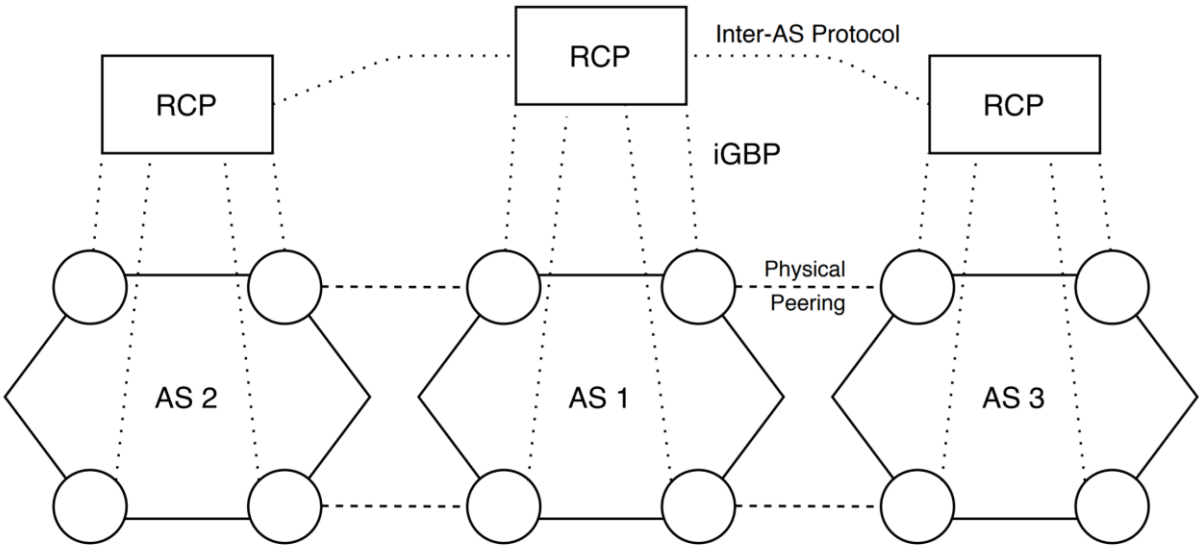

Figure 2. Routing Control Platform (RPC).

The new concepts were not welcomed with open arms and faced a lot of criticism. A lot of researchers felt that the separation between the control and data plane was not a constructive way of moving forward and had fears about the functionality. The wider vision to separate control and data planes led to an exploration of a clean slate architecture for logically centralized control. The 4D project has four layers consisting of the Decision plane, the Dissemination plane, the Discovery plane, and the Data plane. The decision plane holds the logic which controls the network-wide view of the topology and the discovery plane acts as a data provider regarding measurement data and providing network-level objectives. The decision plane controls the data plane. The 4D project had several advantages such as separating networking logic from distributed system issues, as well as providing higher robustness, better security, and means for innovation and evolution of the network architecture [17].

The Ethane Project initiated a simple flow-based ethernet switch that is controlled centrally and it was backward compatible with existing switches and hosts [18]. The initiation of the Ethane project led to the creation of OpenFlow API [15]. OpenFlow is one of the most renowned protocols for SDN. The introduction of OpenFlow struck the perfect balance between a fully programmable network and pragmatism that can lead to real-world implementation. OpenFlow provided a platform of an open protocol that can program the flow tables in different switches and routers [19].

2.2 Architecture of SDN

The main advantage of the SDN is the concept's ability to separate the control plane and data plane. This architecture allows for central control of the network system and its behaviour [20]. The control of traffic and traffic data was tightly coupled with hardware devices in the traditional network. Alongside this, the functions were distributed across many hardware

devices. There are limitations in traditional networks such as management complexity, unscalability, and vendor dependency [21].

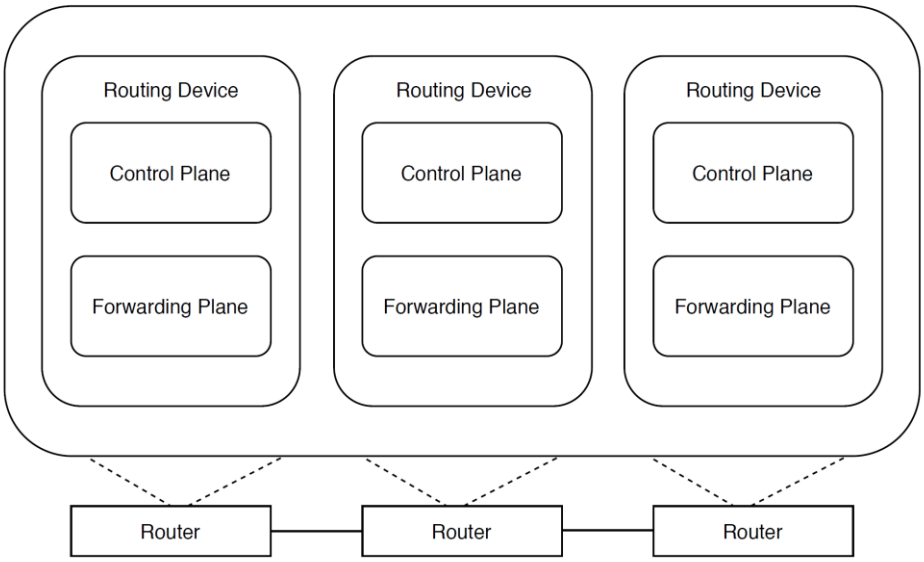

Figure 3. Conventional Network Architecture.

Figure 3 shows the architecture of a conventional network. The forwarding plane and control plane are mounted on the same plane. The networking system is rooted in fixed-function routing devices or switches. The whole architecture is implemented with dedicated devices consisting of one or more routing devices and switches. The limitation of conventional networking architecture is the use of an application-specific integrated circuit (ASIC). This limits the functionality of the hardware. The SDN architecture shown in Figure 4 allows the decoupling of control and data planes. The architecture is divided into three main layers: The Infrastructure Layer, the Control Layer, and the Application Layer. The layers are interconnected with APIs. The control layer in the middle has a global view, the southbound API connects the control layer with the infrastructure layer, and the northbound API connects the control layer with the application layer.

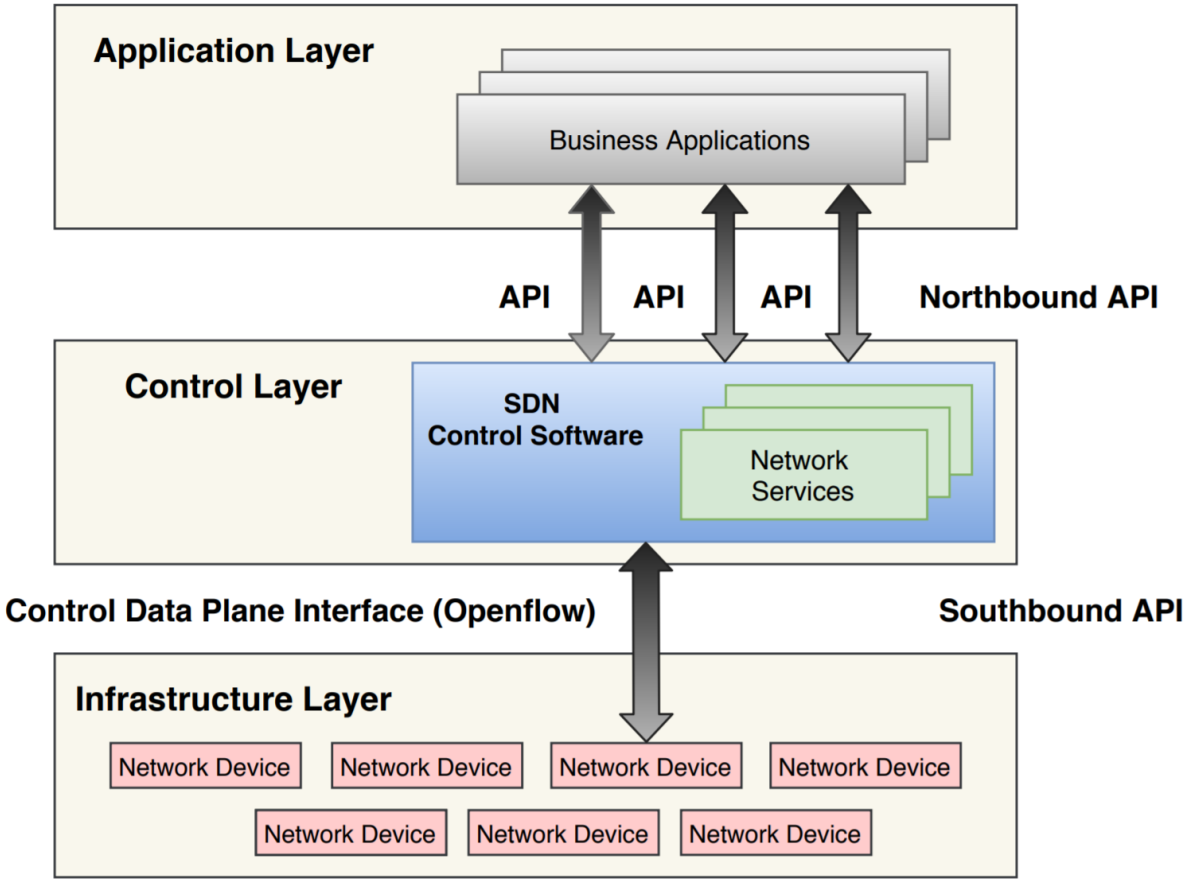

Figure 4. SDN Architecture.

The Infrastructure Layer consists of different switches and routers. Figure 5 shows the main difference between the infrastructure layer and the traditional network architecture is that this layer does not have control capability. Instead, control belongs to the centralized controller. The new infrastructure is built on the basis of open and standard interfaces which is very critical for configuration and communication between the hardware and the software. The controller and the forwarding devices are very important elements in SDN architecture [22]. The forwarding elements consist of flow tables and have matching rules, instructions, and statistics [23]. OpenFlow is one of the most used designs for SDN data plane services even though there are several other forwarding services such as Protocol-oblivious forwarding (POF) [24], Negotiable Data Path Model (NDMs) from Open Network Foundation's (ONF) Forwarding Abstractions Working Group (FAWG) [25].

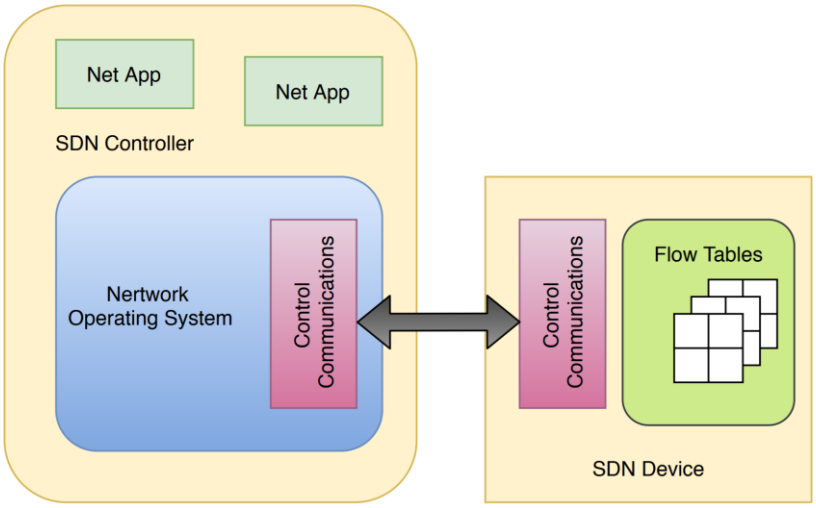

Figure 5. Infrastructure Layer.

The Control Layer in Figure 6 is the most important layer of the SDN architecture. The controller inside this layer controls all the communication in the infrastructure layer. The control layer mainly has two APIs called the northbound API and the southbound API. There are also the westbound API and the eastbound API which are responsible for the connection between several controllers. There are several controllers available such as POX, NOX, Floodlight, OpenDaylight, Flowvisor, Ryu, Open Network Operating System (ONOS), and so on [23]. The control layer is responsible for the commands and flow towards the network devices. The communication between switches and the controller happens through the protocol (e.g. OpenFlow) via the SBI. It is also responsible for topology management and services [26]. These can be modified with different supported features. The controllers which are mainly employed in commercial and development use are mentioned in Table 1.

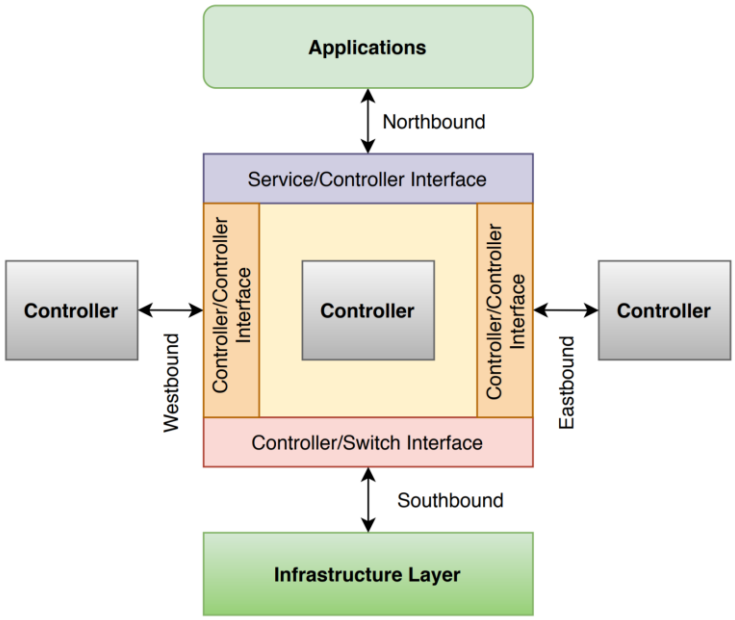

Figure 6. Controller Layout.

Table 1. List of some SDN controllers

Controller	Language base
POX/NOX	Python
Floodlight	Java
Ryu	Python
OpenDaylight	Java
Open Network Operating System (ONOS)	Java
Beacon	Java

The Application Layer defines the features, services, and policies. The application layer is responsible for making the dynamic changes when there are any changes in the network system such as policy change, topology change, and so on [27]. The application layer is also responsible for traffic engineering, such as load balancing, routing, traffic optimization, and traffic measurement and monitoring. The application layer gives the opportunity for innovation by using the network information, network topology, network state, network statistics, and so on.

The layers of the SDN architecture are connected with NBI and SBI. These are also some of the main components of SDN architecture. The northbound API works as the link between the controller and the applications. There are a variety of different northbound API for the SDN and they can control different types of applications through the SDN controller. The optimization of the network, including load balancing, firewalls, and security services, is orchestrated with this interface [28]. The southbound application program interface is the connection between the controller and the forwarding devices. The SDN controller is responsible to manage switches through the southbound API. OpenFlow is the most often used southbound API and it defines a set of commands for the forwarding data. These commands help routers to discover the topology of the network and behavioural definition of physical and virtual switches. These are all based on the application request sent through the northbound APIs. The interface is very crucial for the separation of control and data planes of the SDN environment. Apart from OpenFlow, there are several proposals for the SBI such as ForCES, Open vSwitch Database, POF, OpFlex, and so on [22].

2.3 OpenFlow Protocol

There are different communication protocols that can address the forwarding plane of the networking elements inside SDN. A few of the most common ones are OpenFlow, ForCES, POF, and so on. OpenFlow is the most widely used one and the most popular one. For that reason, we want to discuss it in detail. OpenFlow is a communication protocol that works through the southbound API of SDN and enables controller-to-switch communication. Figure 7 shows the protocol having some forwarding and management functions that allows the programmers to address network operations by routing packets. It allows the control plane to be abstracted from the forwarding plane. It was designed at Stanford University and is currently managed by the Open Networking Foundation (ONF) [29]. The open standards of the OpenFlow protocol allow different devices from different vendors to be connected and managed under a single OpenFlow capable controller [30]. OpenFlow protocol sets up the definition for the specification that needs to be followed for communication between the OpenFlow controller devices and the OpenFlow enabled switches. This communication is performed through a secure channel where the OpenFlow controlling messages as well as the data packets move from controller to switch and the other way round. The OpenFlow protocol

mainly deals with a concept of switches having a forwarding table and an open API that is supported by an OpenFlow based controller.

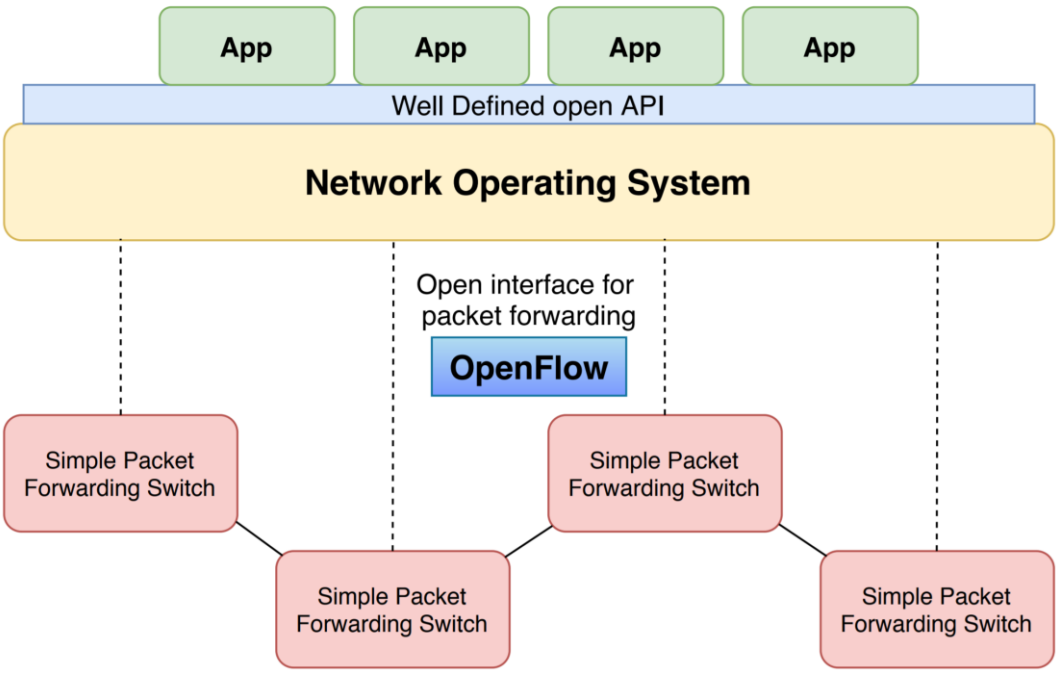

Figure 7. OpenFlow Protocol.

An OpenFlow switch mainly directs communication between the controller device and the switch through the OpenFlow protocol. An OpenFlow switch specification is illustrated in Figure 8, and it is the combination of three important components: installed flow tables in switches, a controller, and OpenFlow protocol which allows the controller to communicate with the switches. An OpenFlow entry in the forwarding table has the match fields, the statistic fields, and sets of instructions which are applied in the packets. After the arrival of a new packet, it will be checked for the matching fields of the flow tables. The actions are performed after that. The flow entries have basic actions for incoming packets and start with forwarding of packets to switches, encapsulating and sending them to the controller with a secured link, dropping packets if required, and also modifying and sending packets to the normal processing pipeline.

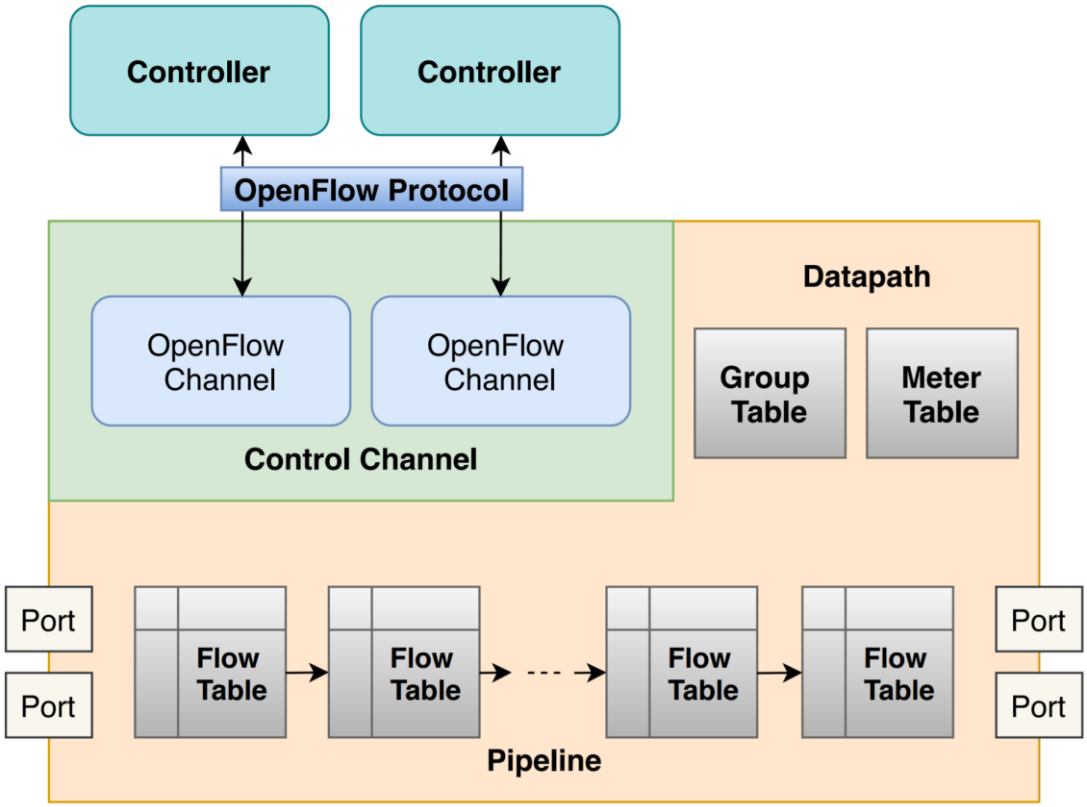

Figure 8. OpenFlow switch specification.

OpenFlow network architecture has three basic models,

1. OpenFlow supported switches in the data plane.
2. The control plane having one or more controllers.
3. A secure channel linking the switches with the control plane.

Communication between the switches as well as with the hosts is done through the data path. Communication between the controller and the switches is done using the control path. Secure connections are maintained through SSL and TSL cryptographic protocols. Suitable security practices are always needing to be implemented to defend from different attack. The connection between the controller and the switch is known to be OpenFlow channel and the OpenFlow protocol is the message pattern for the communication between the controller and the switches. OpenFlow protocol started its development in 2007 and the first version was released in 2009, OpenFlow 1.0. Multiple flow tables were inserted in the next update in version 1.1. The major milestone update came along in version 1.3 and that one is widely used [30]. The latest version was OpenFlow version 1.5.1 from ONF [31].

2.4 SDN Controllers

The controller in the SDN is the core of the network system. The SDN controller is the block of software that controls and manages all the infrastructure between the applications [32]. As mentioned earlier, OpenFlow is the most common protocol for the SDN controller and allows it to direct switches of the packet flow in the network as well as managing different tasks. Figure 9 shows the core functionality modules alongside the interface and some common applications

including the controller, switch management unit, packet processing unit, device manager, topology manager, routing, OpenFlow implementation, service abstraction layer and plugins, and management interface.

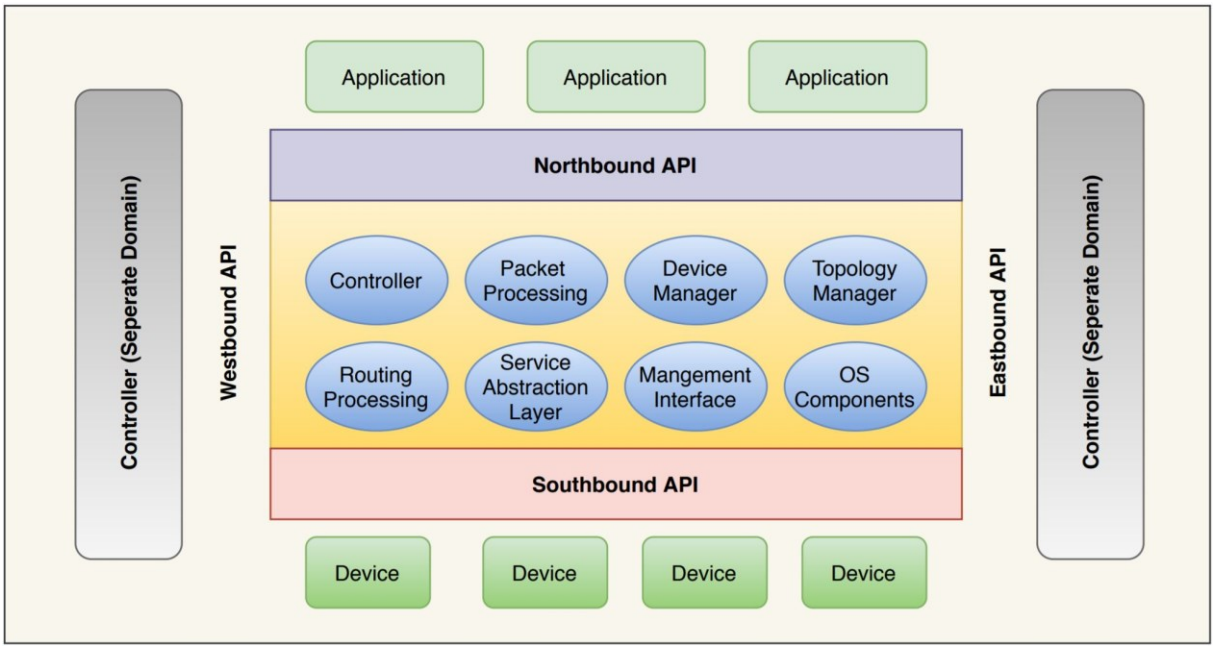

Figure 9. Core Components of SDN.

There are several controllers for the SDN. Different controllers provide different features and approaches. The programming language also plays a role in choosing the controller. The most commonly used controllers for research work are NOX, POX, Floodlight, and Ryu controller. The programming language, good library, and ease of use are some of the reasons for these controllers to be used more often.

2.4.1 NOX Controller

Networking Operating System is known as NOX for short, and it is the first OpenFlow controller. It was developed and released by Nicira Networks [33]. NOX is built on the event-based programming model. It holds the foundation of the SDN innovation as it is the primary OpenFlow controller. NOX provides a full view of the network and it has the capability of operating in pro-active and re-active mode [34]. In pro-active mode, it utilizes the topology's information to pre-load the forwarding tables. In the re-active mode, the first packet is sent by the switches for each new flow to the controller and the controller then creates the flow table by observing the topology. NOX's programming interface is confined within events, namespace, and network view [35]. NOX is also used for component-based framework for SDN application development. NOX's use is often seen in academic work and research for SDN development. SANE and Ethane are some very popular application for NOX. One of the most important sectors of research with NOX has been power management in a network and home networking [35]. Specific support modules for OpenFlow is provided by NOX and the core provides some assisting methods as well as APIs for interaction with OpenFlow switches. Other components such as host tracking and routing are also available as shown in Figure 10.

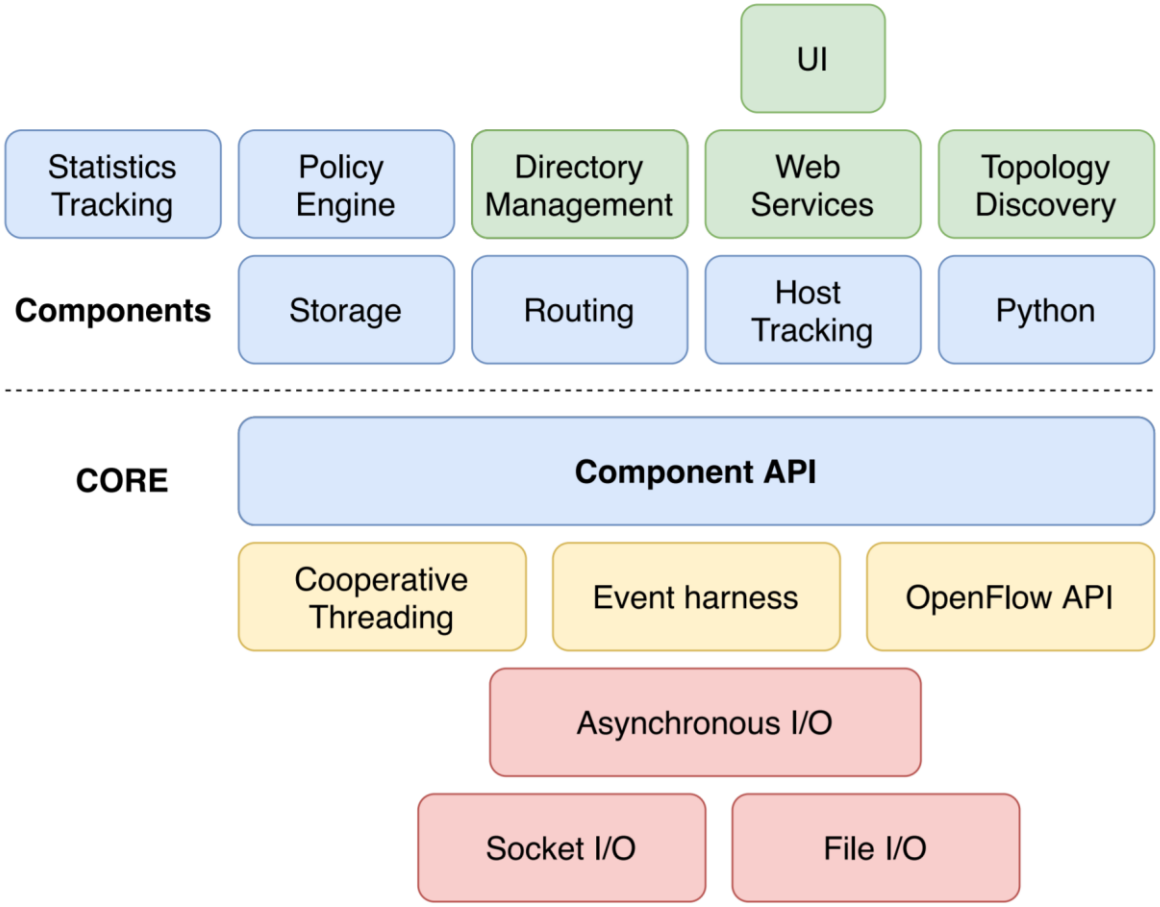

Figure 10. NOX Architecture.

2.4.2 POX Controller

Pythonic Networking Operating System, known as POX for short, is an open-source Python-based SDN controller [36]. It is the newer and upgraded version of the NOX controller. It supports virtualization. It has a simple architecture and the communication between controller and switch uses OpenFlow protocol as shown in Figure 11. The switches act as forwarding devices and perform actions when getting instructions from the controller. Each switch has its own flow table, and, at the start, those are empty. The arrival of packets sends a message of Packet-in to the controller as it gives the instruction to the flow table regarding handling of the packet. POX uses less memory space but has lower throughput compared to other controllers. The advantages of this controller are the Pythonic OpenFlow interface, reusable sample components, and its bundling with Mininet. It supports the OpenFlow v1.0 switches as well as the Open vSwitch [37]. The Graphic User Interface (GUI) with visualization tools makes it easier to use. The POX controller has applications such as hub, switch, load balancer, firewalls, and so on. Our work was done with the POX controller because of the ease of use with the Mininet environment and the Python support.

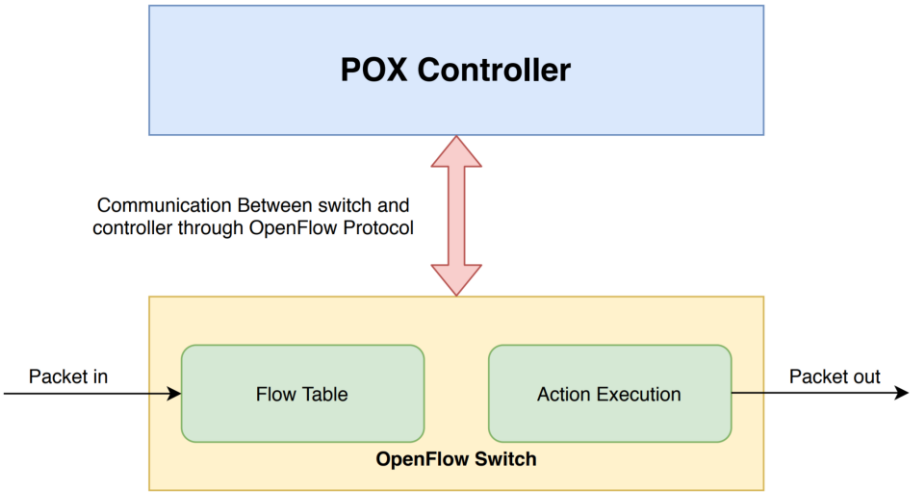

Figure 11. POX Architecture.

2.4.3 Floodlight Controller

Floodlight is a popular SDN controller created by Big Switch Networks. It is an OpenFlow controller. The purpose of the creation of this controller was to improve the functionality of the SDN environment [38]. The core architecture of Floodlight as shown in Figure 12, is modular and has components such as topology management, device management, path computation, generalized storage abstraction, and so on. The components are loadable services. It also has extensive REST APIs with an event notification system. As an open-source controller, it provides developers with the opportunity to develop through software adaptation with Java. The core module known as Floodlight Provider creates an event-driven asynchronous application framework. It is also supported by Mininet.

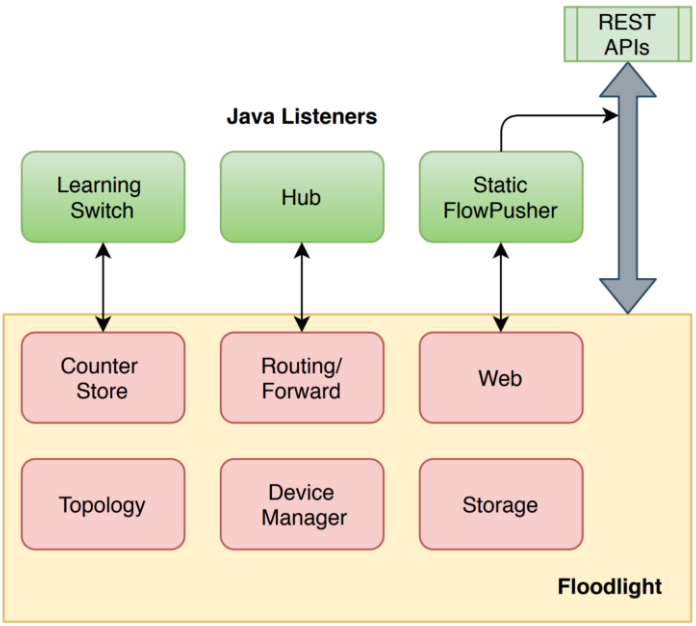

Figure 12. Floodlight Architecture.

2.4.4 Ryu Controller

Ryu controller is an open-source SDN controller based on Python and it is also a component-based controller. It is released and supported by NTT Labs [39]. As shown in Figure 13, it has components such as OpenFlow wire protocol, event management, application management, infrastructure services, and different libraries. It also supports OpenFlow protocol's latest version 1.5. As the controller provides support for well-defined software components and application program interfaces, it creates a bigger scope for developers to widen their research. It allows organizations to have their own customizable deployments according to their specific needs. The code of the Ryu controller is under the Apache 2.0 licensing and open to use for anyone. Even though it supports high availability through the Zookeeper component, it does not yet support the cooperative cluster controller.

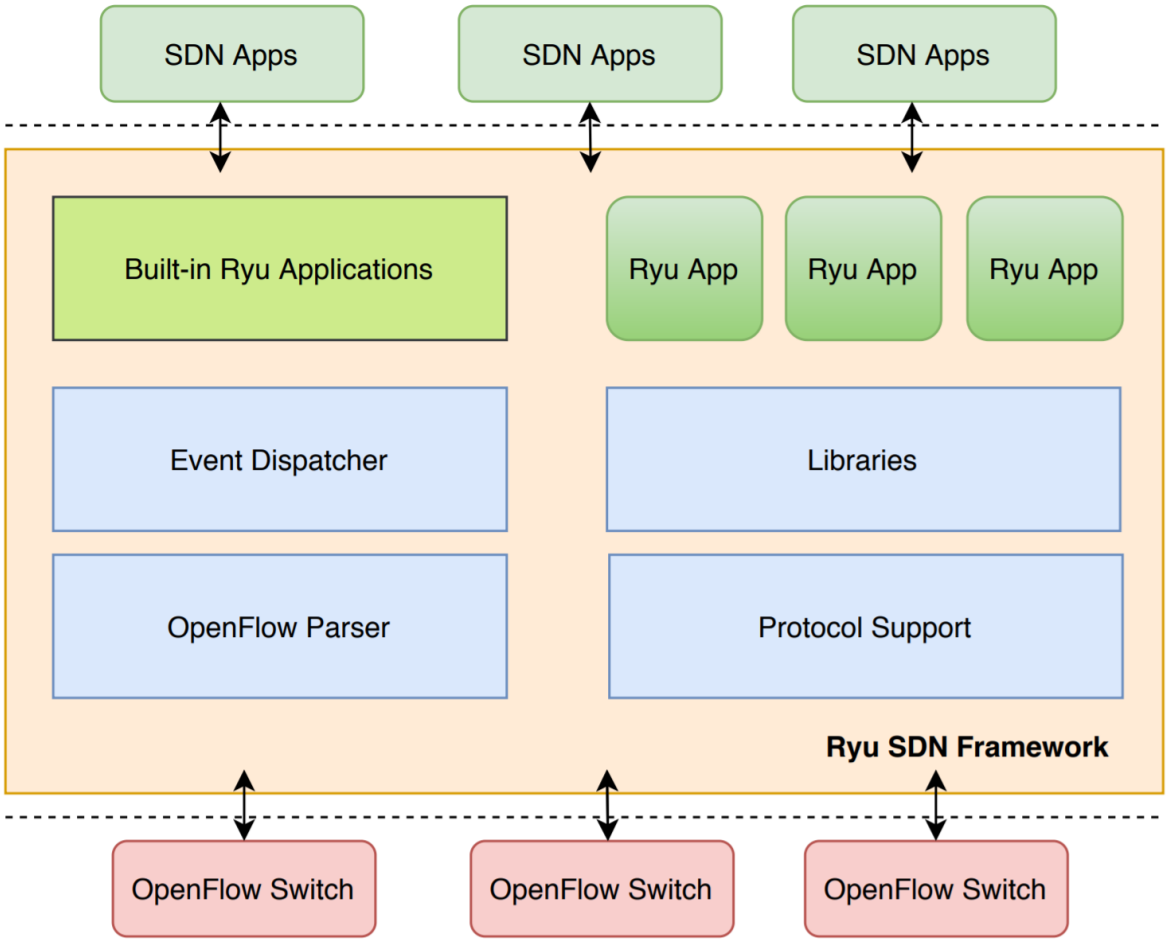

Figure 13. Ryu Architecture.

2.5 Impact of SDN

SDN came with a view to solve legacy networking problems [40], [41]. The SDN enabled environment allows users to go through several solutions to improve the networking environment. The standardization organizations started to move towards SDN and focus more on it. The most relevant one is the Open Networking Foundation [42]. It is a non-profit industry

consortium and includes more than 100 company-members from telecom and network service provider industries, as well as equipment vendor suppliers. It has the vision to make SDN the norm of the networking industry. Other driving forces such as Internet Engineering Task Force (IETF), Software-Defined Networking Research Group (SDNRG), and Interface to the Routing System (I2RS) are working on making SDN the main protocol for the networking environment.

Other organizations such as the Optical Internet Forum (OIF), The Broadband Forum (BBF) and the Metro Ethernet Forum (MEF) are keen on applying SDN principles. Apart from standardizing the platform, there are several research communities as well as summits trying to push this platform forward. The Open Networking Summit (ONS) is one of the biggest summits on SDN. One of the biggest leaps took place when Google entered the SDN structure through its B4 Network [43]. The centralized control of the network made it efficient and flexible. Another SDN system underlying Google's cloud is Andromeda [44]. Other networking companies have been showing interest and rapidly adapting the SDN platform. SDN is trying to solve the stagnant behaviour of the legacy network. Its benefit paves the way for scalable, more controlled, and easily adaptable behaviour. It has a lot of security challenges compared to the legacy network but also paved the way for new solutions. Research is ongoing for the new challenges that it brought with its all the advantages. With its open environment, researchers and developers are able to look through a wider view because of this new paradigm.

3 NETWORK SECURITY

As the world is moving forward at a rapid pace, everything is getting connected and the era of the Internet of Things (IoT) is on the rise. Security has become a very important factor in today's network [45], [46]. Whether it is for a company or for a single user, cybersecurity has taken a huge role in today's technology. It has become a mainstream thing in everyone's lives. According to the National Institute of Science (NIST), an intrusion is an attempt to compromise the confidentiality, integrity and availability of a network system or an attempt to bypass the security mechanism of the network system. Then, intrusion detection is the process of observing and monitoring the event within the network system. The goal of intrusion detection is to observe the behaviour and the events of the network and to analyse them for signs of any abnormal behaviour, anomaly, or any suspicious activity, so that it can help to prevent any attack in the network system [47].

The information security of the network is divided into three sections which are confidentiality, integrity, and availability. They are also known as the CIA triad. Having a good balanced secure network is about the balance between these three main areas of confidentiality, integrity and availability [48]. Confidentiality is about the right of access to information only for the authorized party. Information privacy is a very important thing and breaches take place to steal that data. This is commonly achieved through authentication by user ID and password. Integrity is the authenticity of the information. The information should not be allowed to be modified through unauthorized access and it should also be protected from a system crash and network breaks. Availability refers to the accessibility of information whenever needed. In a network system, availability is hampered by a different type of attacks and blocks people from accessing their information.

Even though information security is a big part of network security, it is not the whole thing. There are several other security threats inside the network and an administrator needs to be aware of those threats to defend and protect the network. The security of the network can be in different layers. There are several security layers nowadays. The adoption of the layers depends on the threat, vulnerabilities, and asset type [49]. One model of a security layer is shown in Figure 14.

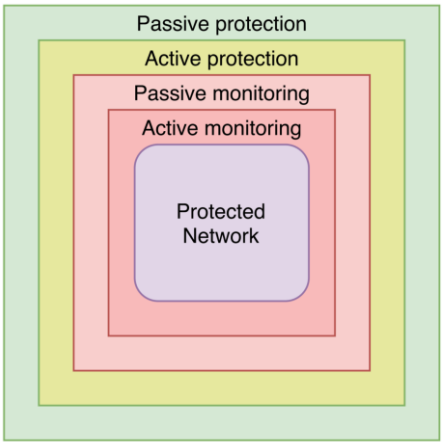

Figure 14. A layered security model.

3.1 Types of Attacks in Network System

A Network system is subjected to different types of attacks. The main two categories are passive and active [50]. There are other types of attack also and a system should be able to properly detect the attack and be able to prevent it.

3.1.1 Passive Attack

Passive attacks try to intrude the system through observed data [51]. It uses unencrypted traffic. The information it gathers is used for several other attacks. Unauthorized personnel's observation of and listening to a network communication channel is labelled as a passive attack. Traffic analysis, decryption of encrypted traffic, and capturing private information are different types of passive attacks.

3.1.2 Active Attack

Active attacks refer to an attempt at unauthorized change to the system. It includes modification, or even creating and sending new data into the system. There are several types of active attacks such as masquerade attacks, message replay, and DoS attacks. These attacks interrupt services and modify data within the network and fabricate the data of the network system [52].

3.2 Defence Techniques

Different attacks are in need of different solutions to monitor, detect and trace them. After the detection of these attacks, there are certain ways to defend against them and prevent damage to the system.

3.2.1 Cryptography

Cryptography is used as a defence mechanism to defend the information in the network system from the attackers. It is a method where the data is sent through the network in an unrecognizable form so that it can only be recognizable to the receiver [53]. It mainly concerns the confidentiality, integrity, and authentication of the data.

3.2.2 Firewall

A firewall is one of the most important security features, and an important line of defence, in today's network. It prevents unwanted traffic from passing through the network and it differentiates a trusted network from a suspected one. These defence mechanisms are created through a different set of rules bound to the firewall in the system [48]. A firewall can be a dedicated device which is specifically designed to filter traffic in the network. It can also be an application that can run in the operating system to block unwanted traffic or application.

3.2.3 Intrusion Detection System

A Network intrusion is unauthorized activity inside a digital network [47]. It is a form of active attack. Normally, it relates to stealing private or valuable information, but it can also be destroying the network system or the data inside the network. As the Internet has grown dramatically in recent years, intrusion in network systems have also increased exponentially. Intrusion detection mainly refers to the process by which unauthorized access is discovered within the network. It is achieved through software that works solely to detect unusual or suspicious activity in the network system [54]. In short, a system that monitors traffic inside the network and detects malicious activity is known as IDS [55]. IDS can be an entire device, or it can be an application running on a network system. It runs on the system to detect abnormal behaviour in the network system. There are two main types of IDS: signature-based and anomaly-based.

Signature-based intrusion detection works with previously set patterns or behaviour. It compares the behaviour of the network, traffic, and actions in the network with the previously set database or patterns. On the other hand, anomaly-based intrusion detection uses learning method which learns the network behaviour and its traffic pattern, and, if it notices something different happening inside the network from its constant normal behaviour, it gives an alert.

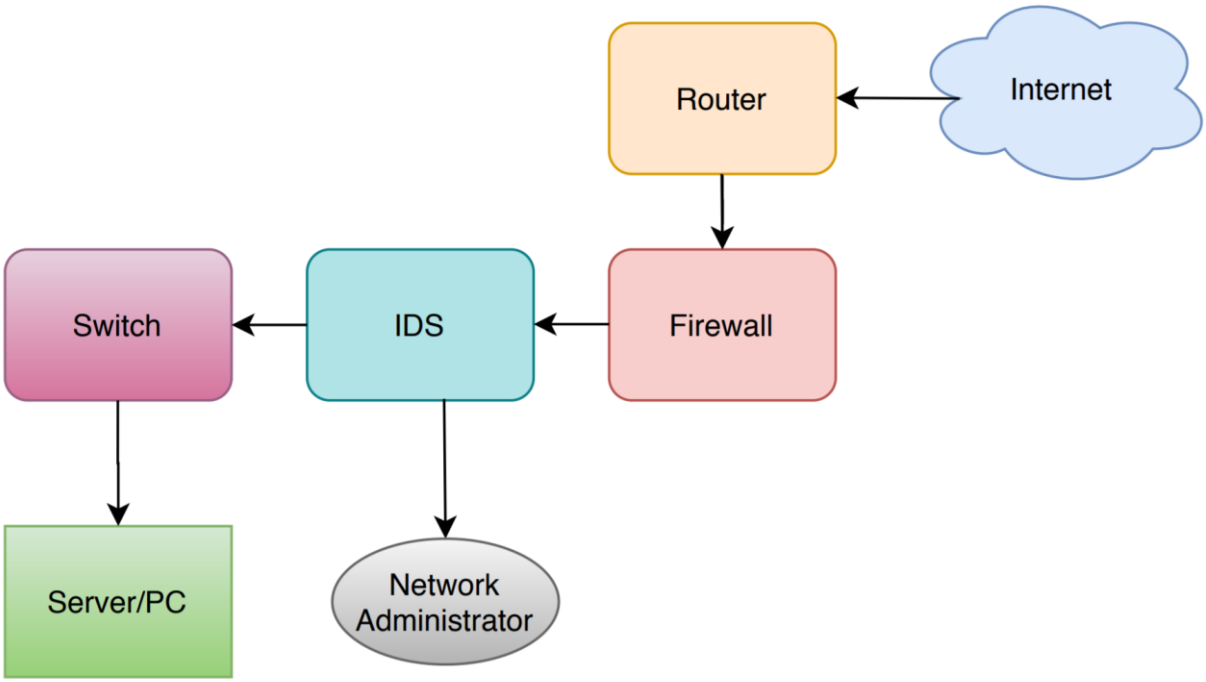

Figure 15. Intrusion Detection System (IDS).

As shown in Figure 15, IDS is just an alert system which alerts the network administrator for abnormal behaviour, attacks, and so on in the network system. The Intrusion Prevention System (IPS) has the functionality of an automatic response towards an intrusion in the network [55]. The focus of this thesis is IDS. There are different types of attacks which IDS detects in the network.

- **The Network Discovery Attack**'s aim is to gather information about the organization from the network. It gathers information regarding the MAC address and service set identifiers (SSID) information. It collects the information for a later attack in the system [56].
- **Eavesdropping** is done to monitor and capture important information about network traffic. It tries to obtain the communication channel as well as the MAC address of both the access point and the client transmitting the data.
- **Masquerading** is done by stealing the data by imitating the characteristics of the client or the access point. It can get the authentication capability in a network and steal the data.
- **A Man-in-the-Middle** attack is done within the communication of two users by intercepting, modifying, and deleting data.
- **A Denial-of-Service (DoS)** attack is done to halt the traffic flow within the network by consuming the network resources or flooding a single host or multiple hosts. This thesis is based on a DDoS attack.

3.3 Network Security in SDN

SDN's logically centralized control provides the opportunity to scale the applications to a new dimension. Even though there are a lot of advantages attained from the new dimension of the network system, such as increased performance as well as programmability, the platform also brought a lot of new security challenges [57], [58], [59]. There are several issues that are related to the SDN system.

- **A Forwarding Device Attack** can be done through the access points or switches through attacks like DoS attacks. It can result in the failure of the network system or create disruption for a certain amount of time.
- **Threats regarding the Control Plane** is one of the biggest issues since SDN is logically centralized. Therefore, an attack in the controller can be the demise of the whole network system.
- **Communication vulnerabilities** are another issue regarding network security in the SDN platform. Even though the communication channel has its own security protocol such as TLS for data-control in the OpenFlow, it can be disabled by the administration and can create the opportunity for man-in-the-middle attacks.
- **Fake Traffic Flows** are one of the challenges in the platform as an attacker can launch fake traffic flows to overflow the channel. It can also be a DDoS attack to overload the resources in the network system.
- **Authenticity** is another issue which is similar to the traditional network system and it can create problems in the network system.
- **Open Programmable APIs** made the platform a success as well as created certain challenges regarding the security issues in the system. It is one of the aspects that must be managed with certain protocols.

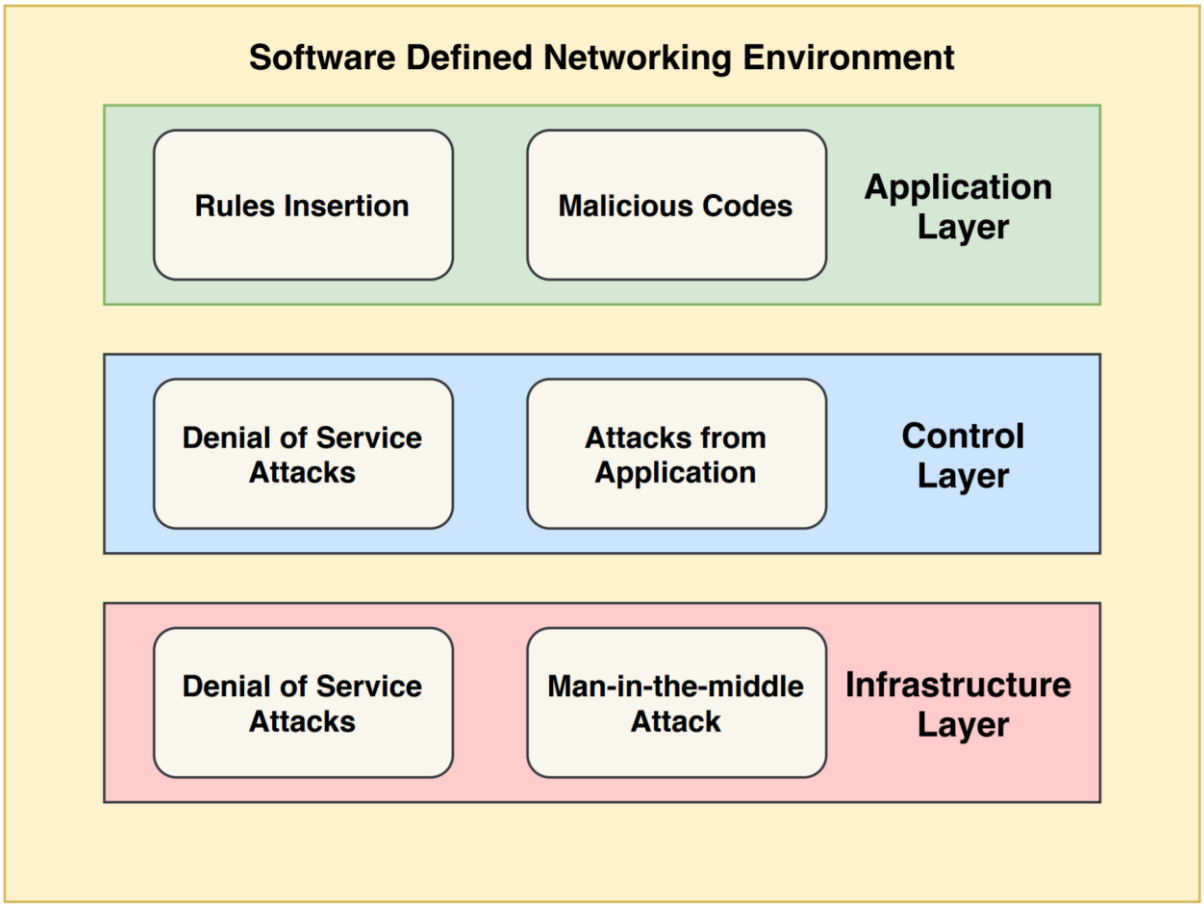

Figure 16. Attacks in different layers of SDN.

Figure 16 shows various attacks on different layers of SDN. For the safer environment of SDN, every layer of the architecture must be monitored and secured [60], [61]. Even though the primary action is to safeguard the controller, other components and layers need to be secured as well [62]. Securing through firewalls, IDS, and IPS is very important and always needs to be dynamically updated.

4 RELATED WORK

SDN has been with us for quite some time and the research to improve the security of the system has been ongoing all the time. Before going through this project's work, it will be valuable to have a look at some work which has been done for the IDS in SDN. The focus is on the IDS, specifically on DDoS attacks. The separation of the control plane from the data plane has made the system vulnerable to this attack, but, at the same time, the platform allowed certain ways of defending against this attack.

DDoS attacks are one of the most dominant problems in the Internet security domain [63]. The DDoS attack mechanism is widely understood, but figuring out a defence mechanism has been quite a challenge as it is hard to differentiate the attack traffic from the normal traffic and the system needs to be updated very frequently. The research for IDS for the DDoS attack in SDN has been ongoing for quite a while and several methods have been tried and tested with several environments with different controllers. The main problem statement is an easy implementation with fewer resources to detect an intrusion.

Mehdi et al. [64], and Giotis et al. [65] discussed various implementation techniques for detecting an attack. Braga et al. [66] used the NOX controller to detect a flood attack. It also utilised the flow-based method for the OpenFlow table, and the main method was monitoring the switches of NOX at predetermined time intervals. The Self Organizing Map (SOM) was used to classify the traffic. By periodically receiving flow table information coupled with the global view of the controller, it can catalogue the flow pattern. The anomaly detector running in the controller determines the attack with the collected information. It works well in a small network but the question arises when it is implemented in a large network system because of the overload in the system for a large amount of flow collection and analysis. Moshref et al. [67], Shin et al. [68], and Vizváry et al. [69] were trying to upgrade the intuitiveness of the OpenFlow switches and focus on the trade-off regarding the resource and accuracy in SDN. There are several methods which are being applied regarding the DDoS attack detection in the SDN. We can investigate those as ML-based approaches and non- ML-based approaches.

4.1 Non-Machine Learning Based Approach

One of the non- ML approaches is the simple statistical approach. The method proposed by Mousavi et al. [70], is an entropy-based approach, in which there is a comparison of the entropy between consecutive packet samples to detect randomness. The calculation was done with a 50-packet window and it is calculated from the destination IP address. An attack is detected if the entropy value becomes less than the threshold value. The proposed method was able to detect an attack in a single host. The question remains regarding its capability to detect an attack in the entire system.

Zhang et al. [71] used the time window of 0.1 seconds and used three different levels of thresholds. The method tried the approach of excluding FP and FN from the network. The method is quite time consuming and uses a lot of resources. No et al. [72] used both packet type and the packet's volume inside the network. It is faster than the previous one but does not mention the amount of resources used in the computation. The method used different datasets to measure the threshold and different entropy values. The FNs are higher in this method, while FPs are lower.

The work of Oshima et al. [73] was based on the work of Mousavi et al. [70], where the author proposed a short-term statistical analysis regarding entropy calculation and used a 50

packet window size. The 50-packet window size was selected after testing several different window sizes. The authors mainly focused on the significance of an attack taking place or not. The method was effective only after the attack traffic was around 75% more than normal traffic. The entropy is measured with the source IP of the incoming packets. The method of Mousavi et al. [70] was tailored specifically for the SDN environment.

Singh et al. [74] has a proposal for a distributed framework and analyses the behaviour of the packet flows. It uses special IP counters and timers to keep track of the network flow and filter the whole process. In this way, it uses the entropy merged with a traceback algorithm to differentiate an attack flow from the normal one. David et al. [75] proposed a similar method with better detection accuracy by analysing network traffic and computing the fast entropy of request per-flow. The author focused on the improvement of the Signature-Based Approach (SBA) and Anomaly Based Approach (ABA). The main problem with regard to the SBA is that it detects only the set of data or signatures that are known threats, and can miss out on detecting attacks from a new malicious threat until it has been put into the database. The ABA faces a bit different problem as it can raise an FP alarm even when the use of bandwidth is legitimate.

There have been several other methods tried for DDoS attack detection such as SPHINX [76] and FloodGuard [77]. SPHINX focused on a framework which uses flowgraph to detect an attack. It is used in real-time and has the ability to detect potentially unknown attacks. FloodGuard focuses on SDN based specific data-to-control plane saturation attacks. It uses a proactive flow rule analyser which runs in the controller. It then uses a data plane cache to migrate the packets using a migration module. This framework adds some overhead delay, but it has a trade-off for accuracy. The work done by Maryam et al. [78] is a variation of the entropy-based method. It uses the destination IP address, flow initiation rate and study of the flow specification. It performs a lightweight attack detection of DDoS attack in the early stages.

4.2 Machine Learning Based Approach

ML has been growing quite rapidly. It is one of the most important aspects of the future generation of the whole technological field. The field of SDN has also seen the implementation of ML in different areas. There are two types of IDS which are signature-based detection and anomaly-based detection and have a different approach. Signature-based IDS has human-created signatures through which the IDS detects malicious traffic. It has high accuracy but can only recognize an attack if the signature is available in the system. Also, the signature-based IDS is quite time consuming while comparing packets with all the signatures to detect attack traffic. On the other hand, anomaly-based IDS uses a statistical method, which has been discussed previously, and it uses deviation of behaviour to detect malicious traffic. It is also a flow-based traffic identification. So, for this project's ML algorithm, we focused on the anomaly-based IDS.

Intrusion detection is defined as a classification task in ML [79]. Supervised ML is applied more often for IDS. The input of different dimensions, such as flow features, has an impact on the performance of ML algorithms. One of the main factors that needs to be taken care of is the detection accuracy as well as processing speed. SDN's capability of centralized logical control with the global view accommodates the ML-based IDS [80]. The global view makes it easier to learn the flow of traffic and react to the attack that happens in the network. Different types of IDS have been researched and studied, such as coarse-grained intrusion detection, fine-grained intrusion detection, and DDoS detection. This project focuses on the DDoS attack.

The DDoS attack has been a major threat from the very beginning. The aim of a DDoS attack is to create an overflow of traffic and exhaust the resources of the network. As SDN is centrally controlled, exhausting the flow of the network while also consuming a high amount of resources leads to unavailability of the network system. Barki et al. [81] implemented an IDS consisting of Signature IDS and Advance IDS. The implementation was done using the Ryu controller. The dataset consisted of records like request time, source and destination host and flag bit. The IDS module classified the traffic as normal and abnormal traffic with algorithms such as k-Nearest Neighbours (KNN), Naive Bayes, k-means, and k-medoids. The authors used the classifier to detect abnormal behaviour and sent it to the advance module to determine whether it is from an authorized user or from a random attacker.

Elsayed et al. [82] benchmarked several algorithms with a public dataset. The observation was done using SVM, Naive Bayes, J48, and Random Forest. The comparison showed that J48 had the highest accuracy although it was only around 80%. The author mentioned the importance of the selection of features and labelling of the dataset. Nanda et al. [83] did something similar to Elsayed et al. [82], by comparing several ML techniques for intrusion detection. The author used C4.5 (Decision Tree), Bayesian Network, Naive Bayes, and Decision Table. It used historical network data for predicting an attack on the host. Average accuracy came out around 91.68% while using the Bayesian Network. The data used in this project was from LongTail project 19 for model training.

Santos et al. [84] also compared different ML algorithms of Multilayer perception (MLP), SVM, Decision Tree, and Random Forest. They used a flow table attack, bandwidth attack, and controller attack. The Scapy tool was used for the attack creation and during the experiment it was found that according to the processing time, Decision Tree was better but with the accuracy, Random Forest came out front. It is to be noted that the author used 11 different features but only 5 of them were critical. The selection of less but the most important features is important as the resource usage is as much important as the accuracy. The models SVM, Naive Bayes, and KNN have been used by Prakash et al. [85]. The author used 6 feature set and trained the model by labelling the traffic. They used the Floodlight controller which is Java-based as mentioned earlier. The accuracy of KNN came out high with around 97%.

Polat et al. [86], compared different ML algorithms based on feature selection. Authors used SVM, KNN, Artificial Neural Network (ANN), and Naive Bayes. The focus is on using ML methods with different feature selection to observe accuracy. As the focus was also to determine the best feature selection, they used some methods to select several features. The usage of the feature selection method improved the accuracy very little, while precision and the F1 score varied quite a bit. The other thing here that also needs to be kept in mind is the use of resources. Having a balance between the use of resources and accuracy is a very important thing.

ML algorithms work better regarding the accuracy of the IDS. They perform better with good resource management. Algorithm selection is important, and, for that reason, the comparison of different algorithms is done. Feature selection is closely related to the algorithm used. The features need to be carefully selected as well as the algorithm. ML uses learning feature behaviour to detect an attack, which gives an advantage over other non-ML methods that focus on the deviation of the behaviour of the traffic. In some circumstances, they can detect high bandwidth usage as abnormal traffic.

5 PROPOSED METHOD

The thesis work is the evaluation of ML methods for detecting intrusions in SDN. The DDoS attack has been a constant threat to both conventional networks and SDN. The DDoS attack is used to consume the resources of the network system. The attacker changes the way of attack with time and the defence mechanism needs to be updated so that it can detect anomalies in the system. It is hard to identify attack with the traditional approach as it relies more on the deviation of the normal traffic flow. Using ML to train the detection system by learning the traffic patterns from new traffic information is more accurate and efficient. We selected four different supervised ML techniques for performance evaluation.

5.1 Machine Learning Algorithms

ML algorithms have been quite thoroughly used in data mining which allows the user to learn about important structural patterns and models from the data in training. The ML algorithm uses a training phase where it learns about a pattern and creates a model. The second phase is the decision-making phase where it uses the trained model to predict a decision from the input data. There are four kinds of learning techniques in ML. Those are supervised, unsupervised, semi-supervised, and reinforcement learning. SDN provided the platform of a programmable logical control making it easier to implement ML algorithms in the networking environment.

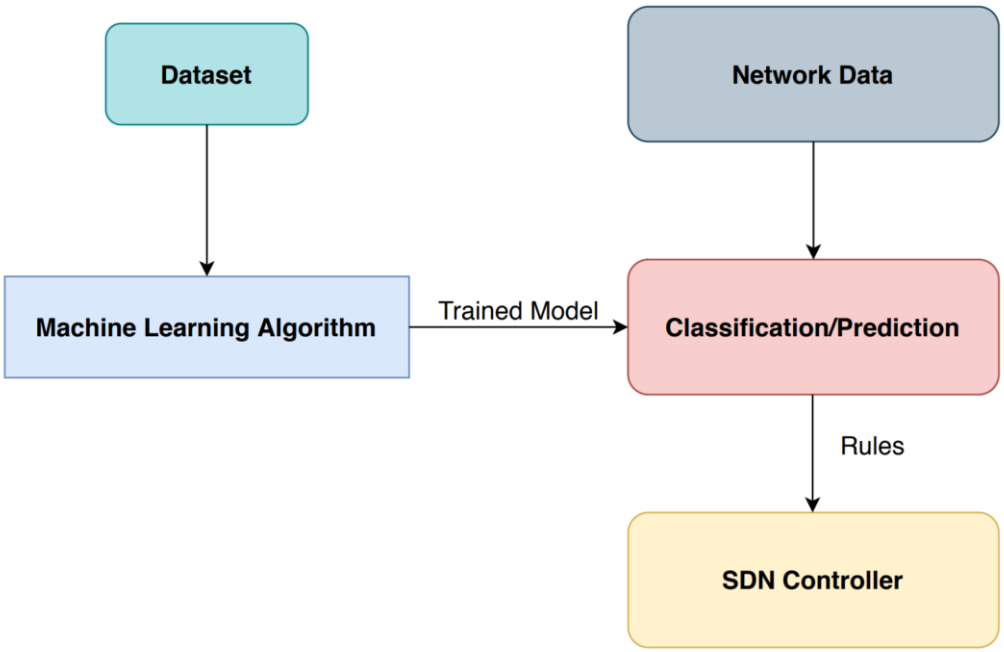

Figure 17. ML process in SDN environment.

Our focus is on DDoS attack detection. ML algorithms have been used in DDoS attack detection. The performance and statistics of our ML approach are measured through accuracy, precision, and error. Figure 17 shows the structure of the implementation of the ML algorithm where the dataset acquired from the traffic flow is fed to train the model to classify network data. We opted for supervised ML algorithms. The ML methods are SVM, Naive-Bayes,

Decision-Tree, and Logistic Regression. We ran these four algorithms and compared the performance between them.

5.1.1 Support Vector Machine (SVM)

The Support Vector Machine (SVM) is a discriminative classifier and it is defined by separating a hyperplane. It is a supervised ML approach. The objective of the SVM is to find a hyper plane that distinctly classifies the data points [87]. Hyperplanes are decision boundaries that help to classify the data points. In a two-dimensional space, the hyperplane separates two sides where each side consists of a distinct class of data. A different kernel function is used to perform mapping. The kernels could be, for example, linear, polynomial, and Radial Based Function (RBF). The classification accuracy depends a lot on kernel selection. The linear kernel used for the dataset is linearly separable otherwise it is separated with polynomial and RBF kernels. The selection of a proper hyperplane is important for good accuracy. The hyperplane which segregates the classes more accurately should be used. The advantage of SVM is that it can learn even with very little data because of its generalization capability. This refers to the ability to use the training data to detect unknown data. In Figure 18, SVM tried several hyperplanes and selected the best fitting one compared to the distance between the two different classes.

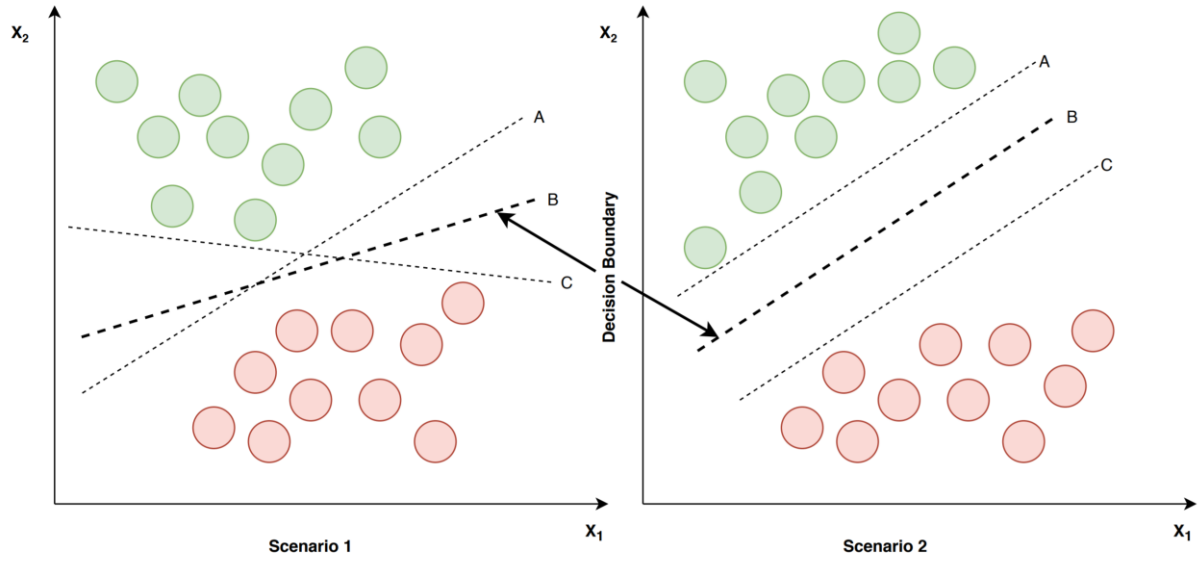

Figure 18. SVM classifier.

5.1.2 Naive-Bayes

The Naive Bayes algorithm is based on Bayes' Theorem with an assumption of being independent among the predictors [88]. The classifier thinks that the presence of a feature in a class is not related to other features that are present. The more data there is in this method, the better the results can be. The training dataset is separated into different class values. Differentiation is created by producing maps of each class value through a list of instances that belong to that class. The complete dataset is then sorted into appropriate lists. The mean and

standard deviation is calculated and conditional probability is used to make the prediction of an attack or normal traffic.

5.1.3 Decision Tree

Decision Tree is one of the supervised learning algorithms which takes decision by learning simple decision rules gathered from the training data [89]. It builds the classification in the form of a tree structure. It divides the whole dataset into small subsets. The final classifier is a result of a decision tree having several decision nodes. A decision node can have two or more sub-nodes. The core objective of the classifier is to pursue the best classification rate. For better resource usage decision tree learning algorithms often build small trees.

5.1.4 Logistic Regression

Logistic Regression assigns observation to a discrete set of classes. It uses the logistic sigmoid function to recur a probability that maps two or more discrete classes. Logistic Regression's prediction is discrete, and its prediction analysis is based on the concept of probability [90].

5.2 Confusion Matrix

The confusion matrix is a way to measure the performance of an ML algorithm. It is an N × N matrix where N is the class. According to our work, we measured two parameters: attack and normal traffic. So, the measurement will be in a 2 × 2 matrix where the column represents the actual class and the row represents predicted class. Table 2 shows the confusion matrix of our work.

Table 2. Confusion Matrix

		Prediction	
		Attack	Normal
Actual	Attack	TP	FN
	Normal	FP	TN

Here,

TP = True Positive, the number of times attack traffic classification was correct.

FN = False Negative, the number of times attack traffic was classified as normal traffic.

TN = True Negative, the number of times normal traffic was classified correctly.

FP = False Positive, the number of times normal traffic was classified as an attack.

We used the confusion matrix to discern accuracy, error, and precision.

Accuracy Rate

This is the ratio of the number of correct predictions and the total number of predictions.

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN}$$

Error Rate

This is the ratio of the total number of wrong predictions versus the total number of overall predictions. It gives the result of how often the algorithm is wrong.

$$Error = \frac{FP + FN}{TP + FN + FP + TN}$$

Precision

Precision is the result of whenever the system makes correct conclusions, that is, how often it is correct.

$$Precision = \frac{TP}{TP + FP}$$

6 SIMULATION AND RESULTS

These experiments are based on an emulation environment using a dataset created from traffic flow within the environment. The reason for this is to observe the traffic and create a dataset, then later to send similar traffic to study detection accuracy. The emulator used is Mininet as it allows the creation of a simulation environment with limited resources. We used OpenFlow protocol which is very popular for SDN research work. We opted for the POX controller and used SVM, Naive Bayes, Decision Tree, and Logistic Regression ML algorithms. The focus was to detect DDoS attacks. The selected attack is User Datagram Protocol (UDP) flooding, which is a common DDoS attack to exhaust the resources of the network system [91], [92]. Below, the details of the experimental environment, experiments, and results are discussed.

6.1 Mininet

Mininet is a network emulator that provides the opportunity to run end hosts, switches, routers, and links on a Linux kernel. The approach of Mininet is to utilize system virtualization features. Its scalability allows the simulation of hundreds of nodes. It allows the user to simulate a large network, of simple to complex architecture, and to simulate the virtual system to visualize the results in a less expensive way [93]. It provides flexibility, applicability, interactivity, scalability, and realistic prototyping [94]. It is easy to create SDN elements and perform the interactions between the elements. The elements such as host, switches, controller, and links can be created through programming or with a GUI element. It creates a complete virtual network and it is easy to perform tasks within the topology where each host acts as a separate individual host. There are several simulators for OpenFlow, but the first open-source simulator was the Mininet. The OpenFlow switches in the emulator work exactly like hardware switches. It is widely used as a testbed to do experiments for SDN. It is based mostly on Python and easy to use. A simple command-line such as “mn –topo single,3 –mac –switch ovsk –controller remote” creates a virtual topology including three hosts with OpenFlow switches in the kernel with three ports, and switches connected through virtual links. It also sets the MAC and IP addresses for each host and configures the switch for remote control. The ease of use, reliability, and scalability made Mininet very popular for research work for SDN.

6.2 Traffic Generator

UDP flood attack which was generated from Scapy is used for the experiments. Scapy is a powerful yet simple traffic manipulator written in Python programming. It can perform several tasks such as attacks, network discovery, probing, and so on. Scapy runs natively on Linux. It can create packets of different protocols and can perform tests such as scanning, tracerouting, probing, unit tests, attacks, and network discovery. We used three types of traffic for the experiment. One is a normal traffic generator that sends normal traffic throughout the network to single or multiple hosts. One attack type is a single victim attack, which attacks only a single host from another host. Another one is a multiple victim attack, which attacks multiple hosts at the same time.

6.3 Simulation Environment

The test setup was done on an HP laptop with an Intel® Core™ i7-8500U CPU @1.80GHz and 1.99GHz processor with 16GB of RAM. We used a virtual machine to create Ubuntu 18.04 for the Linux environment. We used the Oracle VM Virtual box as the virtual machine emulator. Also, the setup had Solid State Drive (SSD). For making the testbed, we used Mininet to create the testbed where one "remote controller" was used with a tree-based topology with a depth of two. The switches were Open vSwitches with OpenFlow protocol. The controller was POX. Figure 19 shows the network topology used for the experiment.

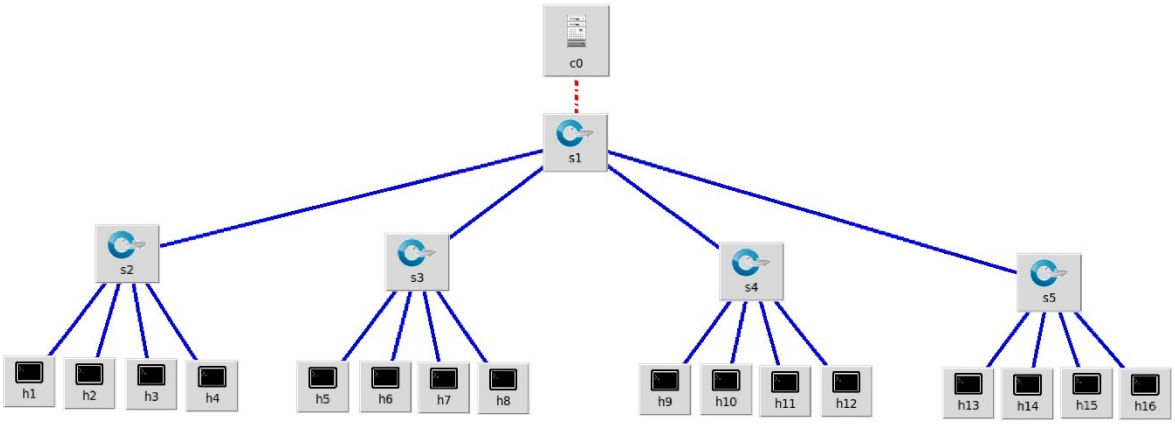

Figure 19. Simulation Network Topology.

6.4 Simulation Process

We used ML algorithms on top of the modified l3_learning module in the POX controller with our ML algorithms. First, we captured traffic flow to create the dataset to use for the ML classifier. We used traffic flow for normal traffic and for the attack traffic which was generated through Scapy and marked the dataset with normal and attack traffic. The features selected were "duration", "Source IP", "Destination IP", "Time to live (TTL) and "IP Protocol". Feature selection is a crucial part of the process as it has an impact on resource usage of the network system.

After creating the dataset, we used the same setup running at modified controller module alongside the ML algorithm at the same time. We used several hosts to create normal traffic as well as attack traffic. The purpose was to analyse the detection of good and bad traffic, as well as, the percentage of good and bad traffic. The controller displays the traffic flow with information on good and bad traffic. The controller was able to identify the good traffic when the normal traffic flow was generated. Later, when the attack traffic was generated, it was able to detect the attack traffic in the network system. Also, the multi victim attack was generated to several hosts. Figure 20 shows that the controller was able to detect and differentiate the attack traffic from the normal type.

```
l3 prototype packet received
packet received
[448.89590215682983, '167.248.175.128', '10.0.0.9', 1, '0']
Good Packet Percentage: 49.1355270496 % Bad Packet Percentage: 50.7529280535 %
Good Packets: 881 Bad Packets: 910


l3 prototype packet received
packet received
[449.02339816093445, '25.102.211.236', '10.0.0.5', 6, '0']
Good Packet Percentage: 49.1638795987 % Bad Packet Percentage: 50.7246376812 %
Good Packets: 882 Bad Packets: 910


l3 prototype packet received
packet received
[449.6061120033264, '84.250.117.250', '10.0.0.16', 17, '0']
Good Packet Percentage: 49.1922005571 % Bad Packet Percentage: 50.6963788301 %
Good Packets: 883 Bad Packets: 910
```

Figure 20. Controller window.

6.5 Results

The heatmap in Figure 21 shows the values and percentages of the confusion matrix. The percentage of normal traffic classified as an attack is in the FP with SVM, Naive Bayes, and Decision Tree having the same results. The prediction of attack traffic was better in SVM over other algorithms. Also, the percentage of detecting attack traffic as normal was less in SVM than the other methods.

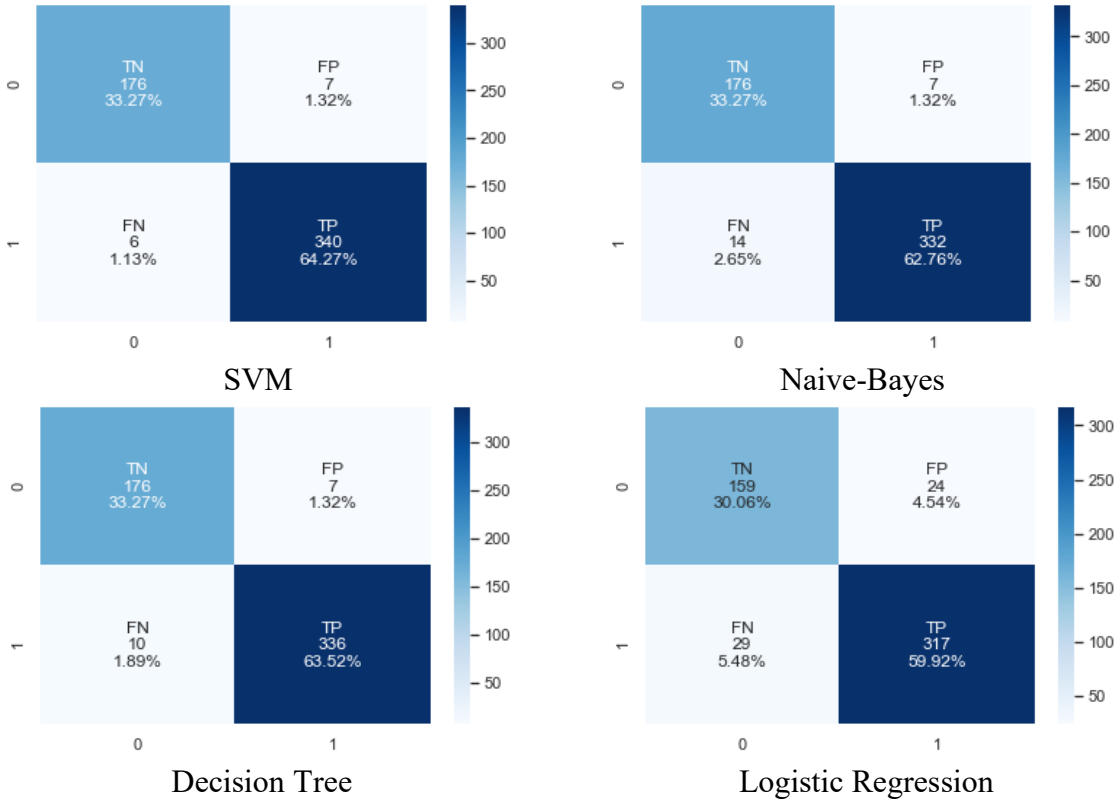

Figure 21. Heatmap of the confusion matrix.

The captured traffic flow during the experiment is shown in Figure 22. It shows the number of packets during certain times per second. The highest peaks are when the attack took place, the last one being the multiple victim UDP flooding. The small peaks in between are around 60 packets per second when there was heavy traffic with a normal flow from several hosts. The region below 30 packets per second is a normal traffic flow from one host.

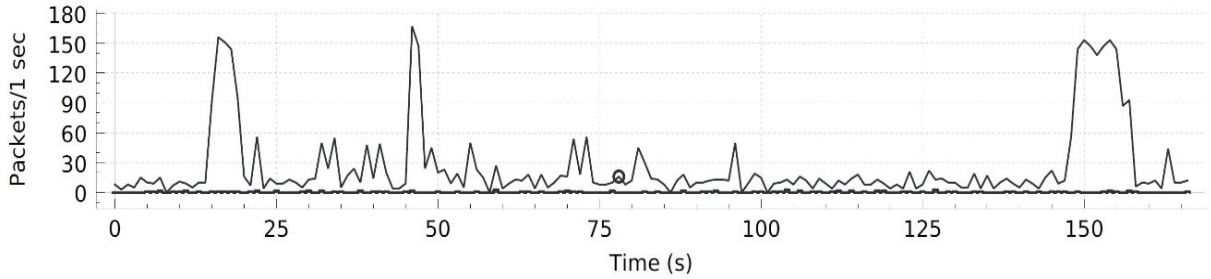

Figure 22. Packet flow during the experiment.

The analysis of the result of our different ML approach is given in Table 3 where it shows accuracy, sensitivity, specificity, precision, and the F1-score. The SVM algorithm performed better in our experimental setup than other algorithms.

Table 3. Performance of different ML models

ML Algorithm	Accuracy	Sensitivity	Specificity	Precision	F1-Score
SVM	97.50%	98.27%	96.70%	97%	96%
Naive-Bayes	96.03%	95.95%	92.63%	93%	94%
Decision Tree	96.78%	97.11%	94.62%	95%	95%
Logistic Regression	89.98%	91.62%	84.57%	85%	86%

In Figure 23, the bar graph gives a visualization of the differences in accuracy and precision of our different ML approaches. From the comparison, we can observe that the SVM method holds the highest accuracy and precision rate among the different algorithms in our experiment. Logistic regression had the lowest accuracy and precision among all of them.

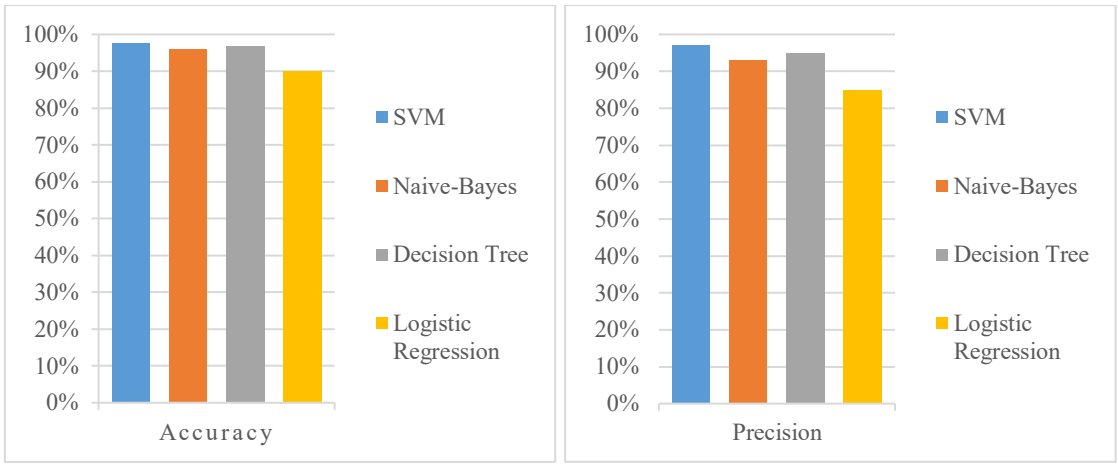

Figure 23. Evaluation of the performance of ML algorithms.

Figure 24 shows the build time differences between the four approaches in the graph. Build-time for Naive Bayes was the shortest while SVM was the longest. The difference of build-time among the algorithms is very low.

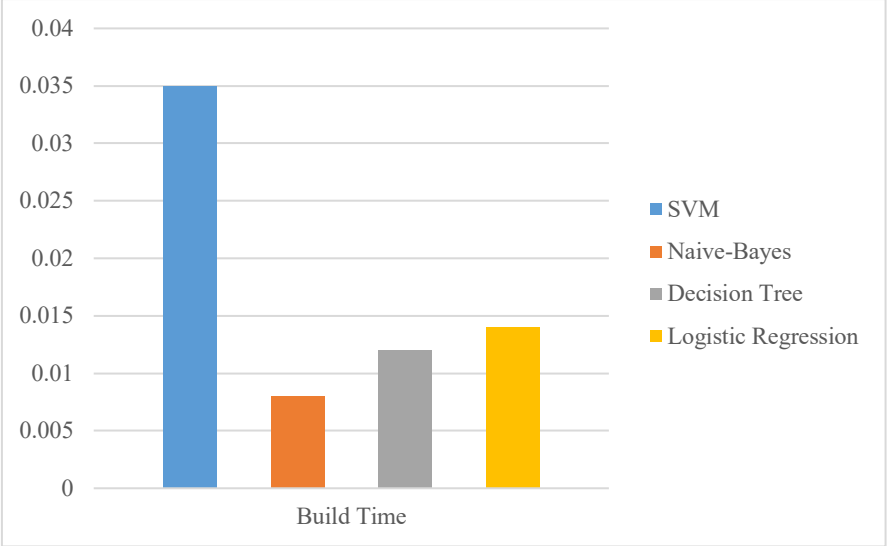

Figure 24. Comparison of the build-time of different algorithms.

The error rate of the algorithms is shown in the graph of Figure 25 below. SVM method had the lowest error rate. Naive Bayes and Decision Tree had less than 4% error while Logistic Regression had the highest error rate among the algorithms in our experiment at almost 10%.

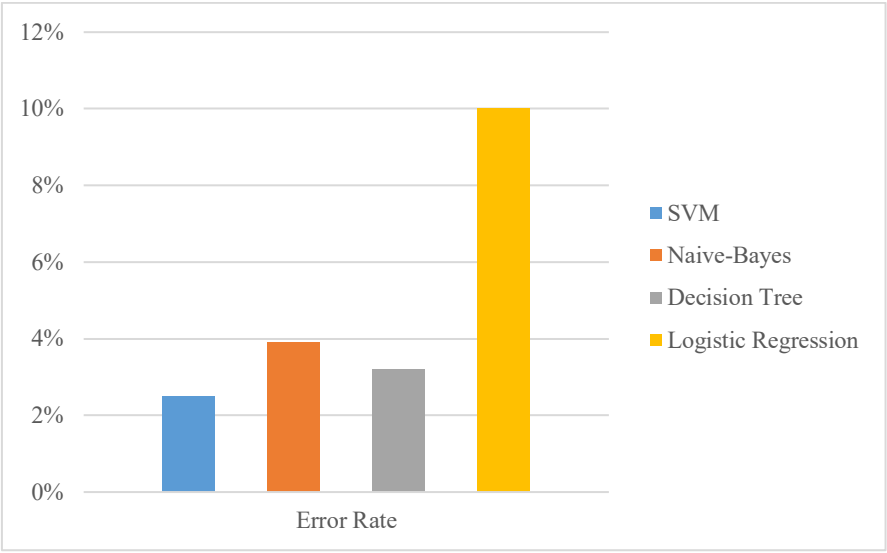

Figure 25. Error rate comparison.

We wanted to compare the algorithms with a non- ML approach. In Figure 26, the SVM algorithm is compared with the work of Maryam et al. [78]. The detection rate for a single victim, and multiple victims through an entropy-based detection system where multiple hosts were generating normal and attack traffic at the same time was 90% and 88.75% respectively.

The accuracy rate is quite good. However, a drawback of entropy-based detection is that it can give an attack warning if there is heavy traffic flow.

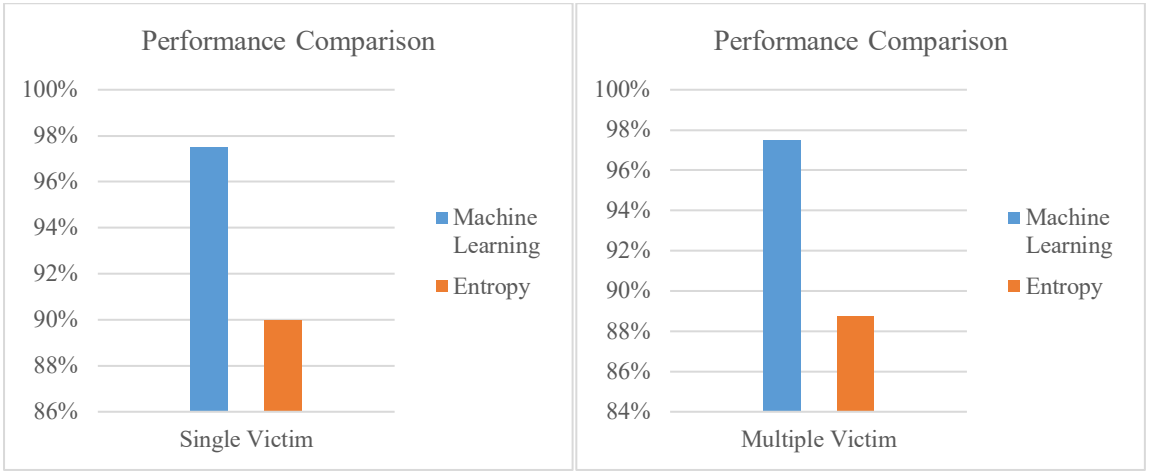

Figure 26. Comparison between Entropy-based and ML-based DDoS attack detection.

A Receiver Operating Characteristics (ROC) curve is a tool that allows evaluation of the test results. Figure 27 is a two-dimensional graph with the TP rate on the Y-axis and the FP rate on the X-axis. The ROC curve is a trade-off between sensitivity (TP rate) and specificity (1- FP rate). Therefore, the algorithm method that has an ROC curve closer to the top left corner indicates better performance. In Table 4, the Area Under Curve (AUC) for different algorithms is given. A good model has an AUC close to 1 which means that it has a good measure of separability. All four models have an AUC over 0.9. Thus, the models have good distinguishability between the two classes of attack and normal traffic.

Table 4. Area Under Curve (AUC) of ML algorithms

ML Algorithm	SVM	Naive Bayes	Decision Tree	Logistic Regression
AUC	0.98	0.99	0.96	0.93

Figure 27 shows the ROC curve of all 4 ML methods implemented in the experiment. From the curves, SVM, Naive Bayes, and Decision Tree show better performance than the Logistic Regression. A range of AUC for the ROC curve between 0.9 and 1 is considered excellent [95]. All the models have an AUC for the ROC over 0.9 while Naive Bayes had the highest value at 0.99.

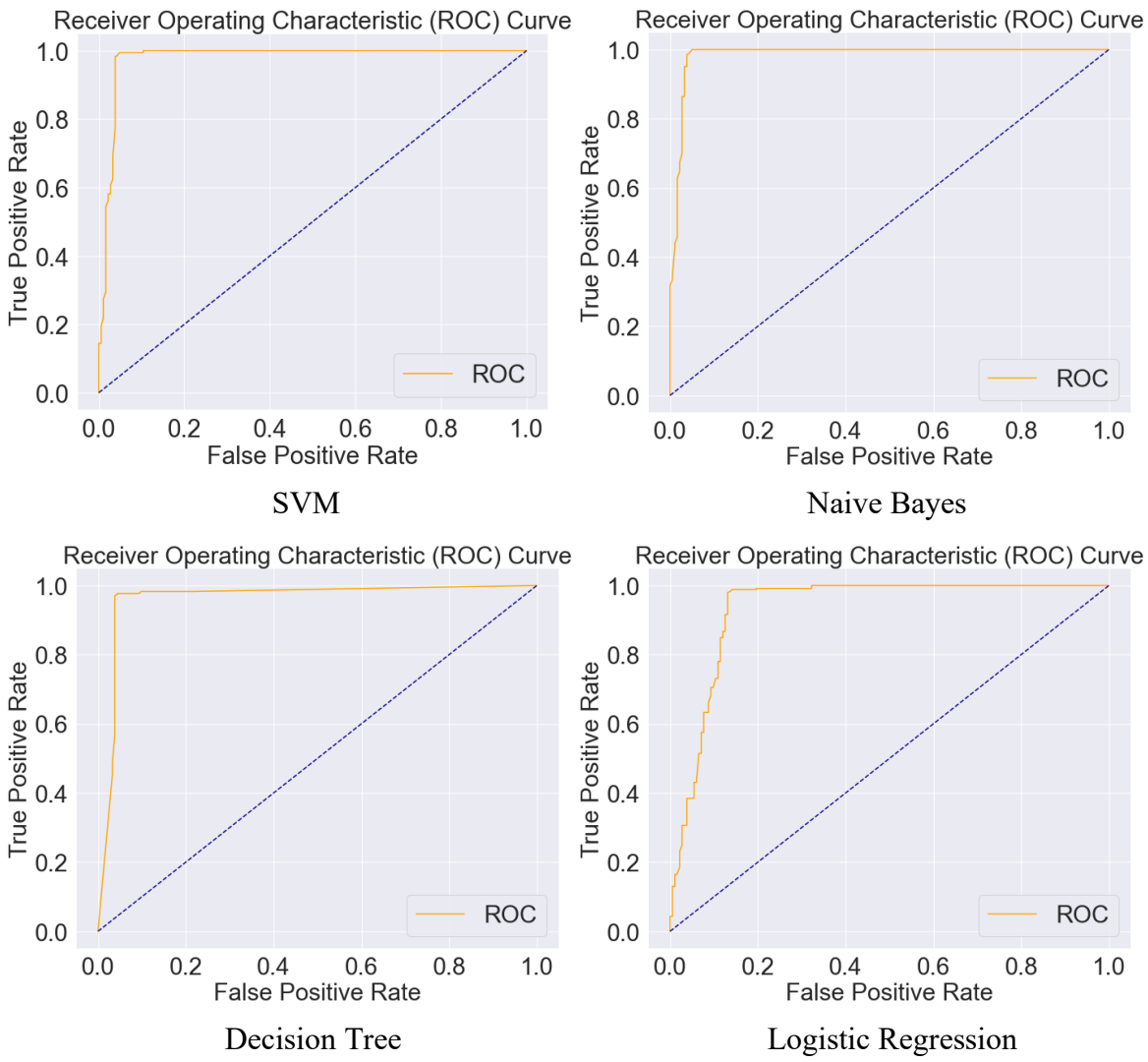

Figure 27. ROC curve of different ML algorithms.

7 DISCUSSION

SDN offers flexibility in a network system which paves the way to overcome challenges from the legacy network. Separating the control and data planes and having a logically centralized control offers the opportunity to develop the architecture and its application easily and in a more efficient way. The centralized and programmable control system brought along new security challenges. In this thesis, we worked on IDS and specifically on DDoS. DDoS has been one of the biggest threats in network security. It exhausts the resource of the network and can sometimes bring down the network system. With centralized control, the threat is higher, and security measures need to be stronger.

The emergence of ML in the SDN field is not new. It has been used in different aspects of SDN. The level of resource usage in a network system is a crucial factor while using ML algorithms [80]. It is one of the factors which relate to feature selection. If more features are used, the ML algorithm takes more resources and time to classify the model. Some of the features in the network flow are crucial for differentiating between an attack and normal traffic. The features selected in our work are the most important ones to differentiate attack traffic from normal traffic. The algorithm selection is also important. The most common selection of ML algorithms has been SVM, Decision Tree, and KNN. To widen the perspective, we selected SVM, Naive Bayes, Decision Tree, and Logistic Regression.

Another important factor is the dataset by which we trained the model. Most of the public datasets are old and have not been updated regularly. We opted to study a model from a dataset generated from the simulated traffic flow within an environment. That dataset was used to train the models and a similar type of traffic flow was generated for analysis of the results. The focus was on real-time model training. The model could generate datasets at certain time intervals to avoid excess resource usage, while keeping the balance of accuracy and resource usage of the network system. Even though we did it manually, it could be automated in the future.

There are different controllers which allow a wide range of options and opportunity for developers. We used the POX controller because of its ease of use. It is based on Python programming language which makes it easier to implement ML algorithms. By placing the ML algorithm alongside the learning module of the POX controller, it was able to run at the same time providing the opportunity to learn and classify traffic more easily. The response time and accuracy were quite a lot better than in the other work mentioned. Only the KNN from [86] had 97% accuracy while most of them had 92% accuracy. Three of the four algorithms used had over 96% accuracy while the SVM attained 97.5% accuracy. The ML algorithms performed better than non- ML algorithms. One of the comparisons was with the entropy-based DDoS attack detection using the same Mininet environment with a UDP flood attack. While looking at the traffic flow curve, it was also observed that the proposed algorithm was not mixing up high use of bandwidth with a DDoS attack.

In our work, we focused mainly on intrusion detection rather than intrusion prevention. Intrusion prevention is based on how well a system can prevent an attack. For future work, intrusion prevention could be implemented. The system could be made dynamic in the future, while the training and classification of the model could be in real-time. As the generated dataset is from real-time simulated traffic flow, the model could be implemented to be trained in real-time to stay updated with the newer attacks.

8 CONCLUSION

As a new architecture with centralized control, SDN provided the ability to manage a network a lot easier than before, but, at the same time, new security challenges came along. The consequences of these security threats can be immense. Different approaches and implementations have been adopted for the IDS such as statistical approach, entropy-based approach, and ML approach. Centralized SDN controller provides the opportunity to implement ML algorithms much easily. The comparison of different algorithm implementations shows different aspects affecting the training, prediction, and use of resources in the network system.

Mininet, OpenFlow protocol, and POX controller were used for the experimental setup. The ML algorithms were SVM, Naive Bayes, Decision Tree, and Logistic Regression. The training dataset was generated from the traffic flow of the experimental network environment. The IDS showed good results as it could differentiate well between normal traffic and attack traffic flow with very little error and a very good accuracy rate. The SVM, Naive Bayes, and Decision Tree performed better than the Logistic Regression, though SVM had the best results among them. Results showed that ML based IDS is more accurate and efficient. The system was only able to detect an attack and the accuracy was good. Means for prevention can be implemented in future work.

9 REFERENCES

[1] "U.S. data breaches and exposed records 2019 | Statista", Statista, 2020. [Online]. Available: https://www.statista.com/statistics/273550/data-breaches-recorded-in-the-united-states-by-number-of-breaches-and-records-exposed/. [Accessed: 25-May- 2020].

[2] Hu, F., Hao, Q., & Bao, K. (2014). A Survey on Software-Defined Network and OpenFlow: From Concept to Implementation. IEEE Communications Surveys & Tutorials, 16(4), 2181–2206.

[3] Ahmad, I., Namal, S., Ylianttila, M., & Gurtov, A. (2015). Security in software defined networks: A survey. IEEE Communications Surveys & Tutorials, 17(4), 2317-2346.

[4] Xu, Y., & Liu, Y. (2016). DDoS attack detection under SDN context. IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications. doi:10.1109/infocom.2016.7524500

[5] Evgeniou, T., & Pontil, M. (2001). Support Vector Machines: Theory and Applications. Lecture Notes in Computer Science, 249–257.

[6] Kaviani, P., & Dhotre, S. (2017). Short Survey on Naive Bayes Algorithm. International Journal of Advance Research in Computer Science and Management. 04.

[7] Rokach, L., & Maimon, O. (2005). Decision Trees.

[8] Maalouf, M. (2011). Logistic regression in data analysis: An overview. International Journal of Data Analysis Techniques and Strategies. 3. 281-299.

[9] Casado, M., Koponen, T., Shenker, S., & Tootoonchian, A. (2012). Fabric: A Retrospective on Evolving SDN. Proceedings of the First Workshop on Hot Topics in Software Defined Networks - HotSDN 12.

[10] Feamster, N., Rexford, J., & Zegura, E. (2014). (4) The Road to SDN: An Intellectual History of Programmable Networks. ACM SIGCOMM Computer Communication Review

[11] Ahmad, I., Namal, S., Ylianttila, M., & Gurtov, A. (2015, July). Towards software defined cognitive networking. In 2015 7th international conference on new technologies, mobility and security (NTMS) (pp. 1-5). IEEE.

[12] Ahmad, I. (2019). Improving Software Defined Cognitive and Secure Networking (University of Oulu, 2019).

[13] K. Calvert. (2006) Reflections on network architecture: An active networking perspective. ACM SIGCOMM Computer Communications Review, 36(2)

[14] Yang, L., Dantu, R., Anderson, T., & Gopal, R. (2004). Forwarding and Control Element Separation (ForCES) Framework.

[15] Caesar, M., Caldwell, D., Feamster, N., Rexford, J., Shaikh, A., & Merwe, J. van der. (2005). Design and implementation of a routing control platform.

[16] Feamster, N., Balakrishnan, H., Rexford, J., Shaikh, A., & Merwe, J. van der. (2004). The case for separating routing from routers. ACM SiGCOMM workshop on Future Direction in Network Architecture.

[17] Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, & H. Zhang. (2005) A clean slate 4D approach to network control and management. ACM SIGCOMM Computer Communications Review, 35(5)

[18] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, & S. Shenker. Ethane: Taking control of the enterprise. In ACM SIGCOMM '07, 2007

[19] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, & J. Turner. (2008) OpenFlow: Enabling innovation in campus networks. ACM SIGCOMM Computer Communications Review

[20] Shah, S. A., Bae, S., Jaikar, A., & Noh, S. (2016). An adaptive load monitoring solution for logically centralized SDN controller. 2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS).

[21] Open Networking Foundation, "Software-defined networking: The new norm for networks." (2012) ONF White Paper.

[22] Kreutz, D., Ramos, F. M., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2015). Software-Defined Networking: A Comprehensive Survey. Proceedings of the IEEE, 103(1), 14-76.

[23] Arlimatti, S., Hassan, S., Habbal, A., & Arif, S. (2015). Software Defined Network and Openflow: A Critical Review: Semantic Scholar.

[24] H. Song. (2013) Protocol-oblivious forwarding: Unleash the power of SDN through a future-proof forwarding plane. 2nd ACM SIGCOMM Workshop Hot Topics Software Defined Networking.

[25] Open Networking Foundation (ONF), ''Charter: Forwarding abstractions working group,'' 2014. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/working-groups/charter-forwarding-abstractions.pdf.

[26] G. V. Nikolaev. (2013) Network Monitoring with Software Defined Networking Towards: OpenFlow Network Monitoring, Faculty of Electrical Engineering, Mathematics and Computer Science, Delft university of Technology.

[27] Albowarab, M. H., Zakaria, N. A., & Z. (n.d.). Software Defined Network: Architecture and Programming Language Survey. International Journal of Pure and Applied Mathematics. 119(18)

[28] Banse, C., & Rangarajan, S. (2015). A Secure Northbound Interface for SDN Applications. 2015 IEEE Trustcom/BigDataSE/ISPA.

[29] Mckeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., & Turner, J. (2008). Openflow: Enabling Innovation in Campus Networks. ACM SIGCOMM Computer Communication Review, 38(2)

[30] Goransson, P., Black, C., & Culver, T. (2016). Software Defined Networks A Comprehensive Approach. Elsevier Science.

[31] OpenFlow Switch Specification Version 1.5.1, Open Networking Foundation, March 2015, [Online].
Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.1.pdf.

[32] Hoang, D. B., & Pham, M. (2015). On software-defined networking and the design of SDN controllers. 2015 6th International Conference on the Network of the Future (NOF).

[33] Jarraya, Y., Madi, T., & Debbabi, M. (2014). A Survey and a Layered Taxonomy of Software-Defined Networking. IEEE Communications Surveys & Tutorials, 16(4).

[34] Shah, S. A., Faiz, J., Farooq, M., Shafi, A., & Mehdi, S. A. (2013). An architectural evaluation of SDN controllers. 2013 IEEE International Conference on Communications (ICC).

[35] Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., Mckeown, N., & Shenker, S. (2008). NOX: towards an operating system for networks. ACM SIGCOMM Computer Communication Review, 38(3).

[36] Fernandez, M. P. (2013). Comparing OpenFlow Controller Paradigms Scalability: Reactive and Proactive. 2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA).

[37] Fancy, C., & Pushpalatha, M. (2017). Performance evaluation of SDN controllers POX and Floodlight in Mininet emulation environment. 2017 International Conference on Intelligent Sustainable Systems (ICISS).

[38] Wallner, R., & Cannistra, R. (2013). An SDN Approach: Quality of Service using Big Switch's Floodlight Open-source Controller. Proceedings of the Asia-Pacific Advanced Network, 35.

[39] Asadollahi, S., Goswami, B., & Sameer, M. (2018). Ryu controller's scalability experiment on software defined networks. 2018 IEEE International Conference on Current Trends in Advanced Computing (ICCTAC), 1-5.

[40] Costa-Requena, J., Santos, J. L., Guasch, V. F., Ahokas, K., Premsankar, G., Luukkainen, S., & Ylianttila, M. (2015, June). SDN and NFV integration in generalized mobile network architecture. In 2015 European conference on networks and communications (EuCNC) (pp. 154-158). IEEE.

[41] Namal, S., Ahmad, I., Saud, S., Jokinen, M., & Gurtov, A. (2016). Implementation of OpenFlow based cognitive radio network architecture: SDN&R. Wireless Networks, 22(2), 663-677.

[42] Open Networking Foundation. (2020, May 20). https://www.opennetworking.org/

[43] Jain, S., Zhu, M., Zolla, J., Hölzle, U., Stuart, S., Vahdat, A., & Zhou, J. (2013). B4: Experience with a Globally-Deployed Software Defined WAN. Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM - SIGCOMM '13.

[44] Enter the Andromeda zone - Google Cloud Platform's latest networking stack. Google Cloud Platform Blog. (2014, April 2). https://cloudplatform.googleblog.com/2014/04/enter-andromeda-zone-google-cloud-platforms-latest-networking-stack.html.

[45] Ahmad, I., Shahabuddin, S., Kumar, T., Okwuibe, J., Gurtov, A., & Ylianttila, M. (2019). Security for 5G and Beyond. IEEE Communications Surveys & Tutorials, 21(4), 3682-3722.

[46] Ahmad, I., Kumar, T., Liyanage, M., Okwuibe, J., Ylianttila, M., & Gurtov, A. (2018). Overview of 5G security challenges and solutions. IEEE Communications Standards Magazine, 2(1), 36-43.

[47] Bace, R., & Mell, P. (2001). NIST special publication on intrusion detection systems. National Institute of Standards and Technology (NIST).

[48] Pfleeger, C. P., Pfleeger, S. L.,and Margulies, J. (2015). Security in Computing. Pearson Education.

[49] Whitepaper: Layered Security: Why It Works, SANS Institute, 2013, [Online]. Available: https://www.sans.org/reading-room/whitepapers/analyst/layered-security-works-34805.

[50] Vacca, J. R. (2017). Computer and information security handbook. Morgan Kaufmann Publishers, an imprint of Elsevier.

[51] Khan, S., Mast, N., Loo, J., & Salahuddin, A. (2008). Passive Security Threats and Consequences in IEEE 802.11 Wireless Mesh Networks. JDCTA, 2.

[52] Pawar, M. V., & Anuradha, J. (2015). Network Security and Types of Attacks in Network. Procedia Computer Science, 48.

[53] Khalifa, O., Islam, M., Khan, S., & Shebani, M. (2004). Communications cryptography. 2004 RF and Microwave Conference (IEEE Cat. No.04EX924).

[54] Tsakountakis, A., Kambourakis, G., & Gritzalis, S. (2007). Towards effective Wireless Intrusion Detection in IEEE 802.11i. Third International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing (SecPerU 2007).

[55] Kemmerer, R., & Vigna, G. (2002). Intrusion detection: a brief history and overview. Computer, 35(4).

[56] Scott-Hayward, S., Natarajan, S., & Sezer, S. (2016). A Survey of Security in Software Defined Networks. IEEE Communications Surveys & Tutorials, 18(1).

[57] Ahmad, I., Namal, S., Ylianttila, M., & Gurtov, A. (2015). Security in software defined networks: A survey. IEEE Communications Surveys & Tutorials, 17(4), 2317-2346

[58] Liyanage, M., Ahmed, I., Ylianttila, M., Santos, J. L., Kantola, R., Perez, O. L., & Jimenez, C. (2015, September). Security for future software defined mobile networks. In 2015 9th International Conference on Next Generation Mobile Applications, Services and Technologies (pp. 256-264). IEEE.

[59] Iqbal, M., Iqbal, F., Mohsin, F., Rizwan, M., & Ahmad, F. (2019). Security Issues in Software Defined Networking (SDN): Risks, Challenges and Potential Solutions. International Journal of Advanced Computer Science and Applications, 10(10).

[60] Liyanage, M., Okwuibe, J., Ahmed, I., Ylianttila, M., Pérez, O. L., Itzazelaia, M. U., & de Oca, E. M. (2017, June). Software defined monitoring (sdm) for 5g mobile backhaul networks. In 2017 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN) (pp. 1-6). IEEE.

[61] Liyanage, M., Ahmed, I., Okwuibe, J., Ylianttila, M., Kabir, H., Santos, J. L., & De Oca, E. M. (2017). Enhancing security of software defined mobile networks. IEEE Access, 5, 9422-9438.

[62] Ahmad, I., Suomalainen, J. & Huusko, J. (2019). 5G -Core Network Security. In Wiley 5G Ref (eds R. Tafazolli, C.-L. Wang and P. Chatzimisios).

[63] Mirkovic, J., & Reiher, P. (2004). A taxonomy of DDoS attack and DDoS defense mechanisms. ACM SIGCOMM Computer Communication Review, 34(2).

[64] Mehdi, S. A., Khalid, J., & Khayam, S. A. (2011). Revisiting Traffic Anomaly Detection Using Software Defined Networking. Lecture Notes in Computer Science Recent Advances in Intrusion Detection.

[65] Giotis, K., Argyropoulos, C., Androulidakis, G., Kalogeras, D., & Maglaris, V. (2014). Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments. Computer Networks, 62.

[66] Braga, R., Mota, E., & Passito, A. (2010). Lightweight DDoS flooding attack detection using NOX/OpenFlow. IEEE Local Computer Network Conference.

[67] Moshref, M., Yu, M., & Govindan, R. (2013). Resource/accuracy tradeoffs in software-defined measurement. Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking - HotSDN '13.

[68] Shin, S., Yegneswaran, V., Porras, P.A., & Gu, G. (2013). AVANT-GUARD: scalable and vigilant switch flow management in software-defined networks. CCS '13.

[69] Vizváry, M., & Vykopal, J. (2014). Future of DDoS Attacks Mitigation in Software Defined Networks. Monitoring and Securing Virtualized Networks and Services Lecture Notes in Computer Science.

[70] Mousavi, S.M., & St-Hilaire, M. (2015). Early detection of DDoS attacks against SDN controllers. 2015 International Conference on Computing, Networking and Communications (ICNC).

[71] Zhang, J., Qin, Z., Ou, L., Jiang, P., Liu, J., & Liu, A. X. (2010). An advanced entropy-based DDOS detection scheme. 2010 International Conference on Information, Networking and Automation (ICINA).

[72] No, G., & Ra, I. (2009). An efficient and reliable DDoS attack detection using a fast entropy computation method. 2009 9th International Symposium on Communications and Information Technology.

[73] Oshima, S., Nakashima, T., & Sueyoshi, T. (2010). Early DoS/DDoS Detection Method using Short-term Statistics. 2010 International Conference on Complex, Intelligent and Software Intensive Systems.

[74] Singh, S., & Jain, S. (2013). A Review of Detection of DDOS Attack Using Entropy Based Approach.

[75] David, J., & Thomas, C. (2015). DDoS Attack Detection Using Fast Entropy Approach on Flow- Based Network Traffic. Procedia Computer Science, 50, 30–36.

[76] Dhawan, M., Poddar, R., Mahajan, K., & Mann, V. (2015). SPHINX: Detecting Security Attacks in Software-Defined Networks. NDSS.

[77] Wang, H., Xu, L., & Gu, G. (2015). FloodGuard: A DoS Attack Prevention Extension in Software-Defined Networks. 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, 239-250.

[78] Maryam, K. (2015). Early Detection and Mitigation of DDoS Attacks in Software Defined Networks (thesis). Ryerson University, Toronto.

[79] Arul, A., Subburathinam, K., & Sivakumari, S. (2013). Classification Techniques for Intrusion Detection An Overview. International Journal of Computer Applications. 76. 33-40. 10.5120/13334-0928.

[80] Khairi, M.H., Ariffin, S.H., Latiff, N.M., Abdullah, A.S., & Hassan, M.K. (2018). A Review of Anomaly Detection Techniques and Distributed Denial of Service (DDoS) on Software Defined Network (SDN). Engineering, Technology & Applied Science Research, 8, 2724-2730.

[81] Barki, L., Shidling, A., Meti, N., Narayan, D. G., & Mulla, M. M. (2016). Detection of distributed denial of service attacks in software defined networks. 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI).

[82] Elsayed, M. S., Le-Khac, N.-A., Dev, S., & Jurcut, A. D. (2019). Machine-Learning Techniques for Detecting Attacks in SDN. 2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT).

[83] Nanda, S., Zafari, F., DeCusatis, C., Wedaa, E., & Yang, B. (2016). Predicting network attack patterns in SDN using machine learning approach. 2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), 167-172.

[84] Santos, R., Souza, D., Santo, W., Ribeiro, A., & Moreno, E. (2019). Machine learning algorithms to detect DDoS attacks in SDN. Concurrency and Computation: Practice and Experience, e5402.

[85] Prakash, A., & Priyadarshini, R. (2018). An Intelligent Software defined Network Controller for preventing Distributed Denial of Service Attack. 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), 585-589.

[86] Polat, H., Polat, O., & Cetin, A. (2020). Detecting DDoS Attacks in Software-Defined Networks Through Feature Selection Methods and Machine Learning Models. Sustainability, 12(3), 1035.

[87] Evgeniou, T., & Pontil, M. (2001). Support Vector Machines: Theory and Applications. Lecture Notes in Computer Science, 249–257.

[88] Zhang, H. (2004). The optimality of naive Bayes. AA, 1(2), 3.

[89] Safavian, S. R., & Landgrebe, D. (1991). A survey of decision tree classifier methodology. IEEE Transactions on Systems, Man, and Cybernetics, 21(3), 660–674.

[90] Peng, C.-Y. J., Lee, K. L., & Ingersoll, G. M. (2002). An Introduction to Logistic Regression Analysis and Reporting. The Journal of Educational Research, 96(1), 3–14.

[91] Gurusamy, U., K, H., & MSK, M. (2019). Detection and mitigation of UDP flooding attack in a multicontroller software defined network using secure flow management model. Concurrency and Computation: Practice and Experience, e5326.

[92] Kolahi, S. S., Treseangrat, K., & Sarrafpour, B. (2015). Analysis of UDP DDoS flood cyber attack and defense mechanisms on Web Server with Linux Ubuntu 13. 2015 International Conference on Communications, Signal Processing, and Their Applications (ICCSPA'15)

[93] Lantz, B., Heller, B., & McKeown, N. (2010). A network in a laptop. Proceedings of the Ninth ACM SIGCOMM Workshop on Hot Topics in Networks - Hotnets '10.

[94] De Oliveira, R. L. S., Schweitzer, C. M., Shinoda, A. A., & Ligia Rodrigues Prete. (2014). Using Mininet for emulation and prototyping Software-Defined Networks. 2014 IEEE Colombian Conference on Communications and Computing (COLCOM).

[95] Bradley, A. P. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern recognition, 30(7), 1145-1159